3-D IS HERE!
(Glasses Included!)

80 micro

1983 USA $5.95

A WAYNE GREEN PUBLICATION

the magazine for TRS-80* users ®

SPECIAL ANNIVERSARY ISSUE

Special 13th Issue:

Over 70 NEW Articles, Including □ Programming Techniques □ Utilities □ Tutorials □ Games □ 2 3-D Stereoscopic Programs □ Disk Utility Pack for Tape Users □ Bill Barden's Assembly-Language Primer □ 200 User Groups Listed □ PLUS a Complete *80 Micro* Annotated Index with Debugs

35
0 74470 65949

M I C R O T E R M

More and more hardware and communications services are allowing speeds up to 1200 baud. Soon, some may be going faster than that. Today's terminal software simply can't keep up. But now there is an alternative. Micro-Systems Software introduces MicroTerm, the high speed terminal.

Model III MicroTerm will communicate, without insertion of null characters, at 4800 baud. Guaranteed. No cop-outs, no question. MicroTerm is so fast that you can exit from the terminal to the main menu, adjust video width, open the buffer, turn on the printer, or any one of dozens of other functions, and return to the terminal mode! without missing a thing!

MicroTerm continues to input from the RS232, even while at the main menu. This is the only terminal capable of such an astounding feat. MicroTerm offers you most of the features that "Brand X" smart terminals have, plus it gives you: • Ultra high baud rate operation (up to 9600 in certain cases). • Input while at menu. • Easy to use translation tables. • Easy to use phone number listings. • Maximum auto dial support — most major brands. • Direct file transfer companion program included at no extra cost (compatible with DFT). • DOS commands from menu without exiting program. • Over 34K of capture buffer (in a 48K TRS-80). • Can be set to automatically dial telephone and transmit buffer at preset time without any operator intervention.

And many, many more great features. MicroTerm is so fast you must see it to believe it. The various menus are displayed so fast, they seem to jump out at you. Status of various functions can be displayed and altered in split seconds.

For the computerist who wants the ultimate, state-of-the-art terminal software, there is no other choice.

MicroTerm retails for $79.95, but registered DOSPLUS owners can purchase it for only $59.95. $20.00 off the retail price! MicroTerm comes complete with the terminal program, the direct file transfer program, some standard translation tables, and documentation.

Don't delay, order yours today! Specify when ordering Model I or III and whether you want it on 40 or 80 track media. Requires a 16K TRS-80 with one disk drive. We recommend 48K for serious communications work. MicroTerm will be available beginning June 30, 1982.

MICRO-SYSTEMS SOFTWARE, INC.

4301-18 Oak Circle
Boca Raton, FL 33431
Telephone: (305) 983-3390

80 Contents

Happy, Happy Birthday 80

Know Your System

Model I/III

80 Micro (ISSN 0744-7868) is published 12 times a year by 1001001 Inc., 80 Pine St., Peterborough, NH 03458, Phone: 603-924-9471. Second class postage paid at Peterborough, NH, and additional mailing offices. Subscription rates in U.S. are $25 for one year and $53 for three years. In Canada and Mexico $27.97—one year only, U.S. funds drawn on a U.S. bank. Canadian distributor: Micro Distributing, 409 Queen St. West, Toronto, Ontario, Canada M5V 2A5. BC Canadian distributor: Graymar Data Services Ltd., #4 258 E. 1st Ave., Vancouver, BC V5T 1A6. Foreign subscriptions (surface mail), $44.97—one year only, U.S. funds drawn on a U.S. bank. Foreign subscriptions (air mail), please inquire. In South Africa contact *80 Micro*, P.O. Box 787615, Sandton, South Africa 2146. All U.S. and Canadian subscription correspondence should be addressed to *80 Micro*, Subscription Department, P.O. Box 981, Farmingdale, NY 11737. Please include your address label with any correspondence. Postmaster: Send form 3579 to *80 Micro* Subscription Services, P.O. Box 981, Farmingdale, NY 11737.

Manuscripts are welcome at *80 Micro*. We will consider publication of any TRS-80 oriented material. Guidelines for budding authors are available. Please send a self addressed envelope and ask for "How to Write for *80 Micro*." *80 Micro* is published monthly by 1001001 Inc., a subsidiary of Wayne Green Inc.

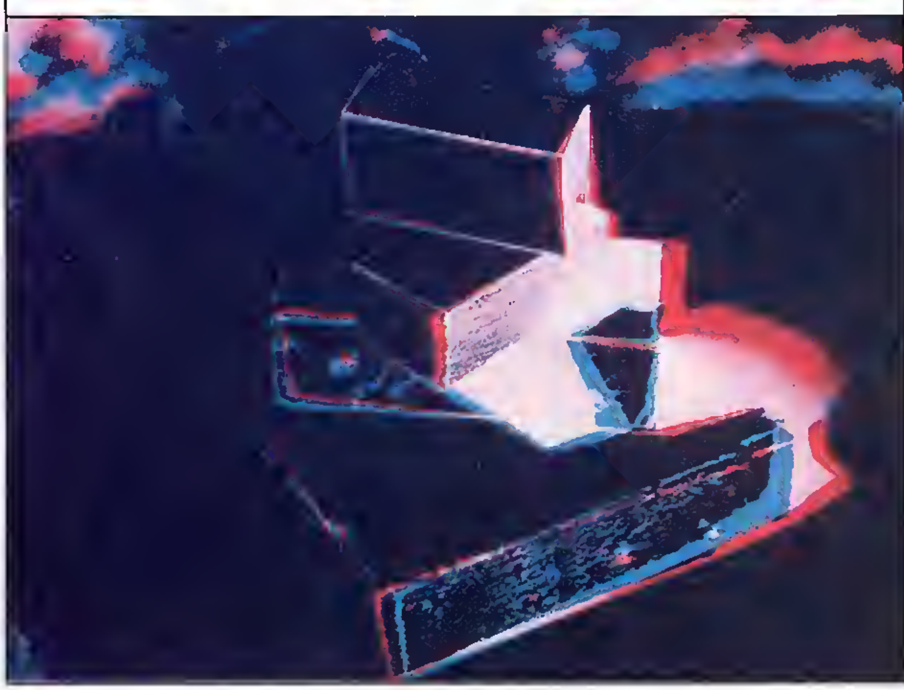
Cover by Ray Maher

***TRS-80, Scripsit and TRSDOS are trademarks of Tandy Corp.**

80 Contents

Color Computer

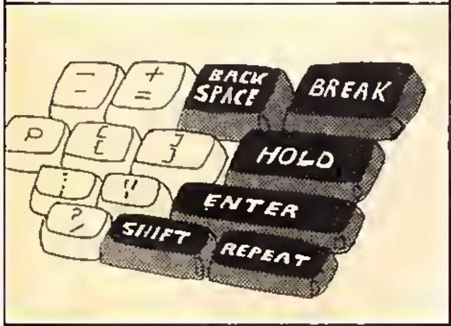
80 Contents



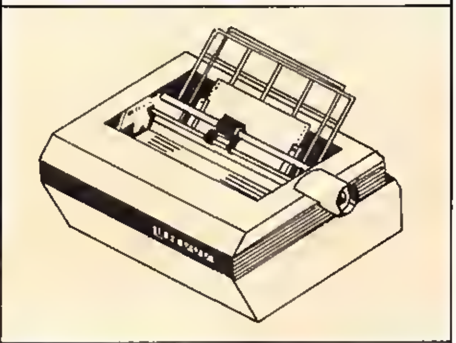
Index

80 Contents

Model II



Peripherals



PUBLISHER/EDITOR
Wayne Green
EXECUTIVE VICE PRESIDENT
Sherry Smythe
ASSISTANT TO PRESIDENT
Matt Smith
GENERAL MANAGER
Debra Wetherbee
ASSISTANT PUBLISHER
Jeff DeTray
CORPORATE CONTROLLER
Roger Murphy
CIRCULATION COORDINATOR
603-924-9471
Patricia Ferrante
BULK & NEWSSTAND SALES MANAGER
Ginnie Boudrieau
ADVERTISING, 603-924-7138
David Schissler: Advertising Manager; Penny Brooks, Mary Hartwell: Sales; Betty Butler: Ad Coordinator.
New England Advertising Representative: John A. Garland, Frank Surace, Garland Associates, Inc., Box 314 SHS, Duxbury, MA 02332 617-934-6464

PRODUCTION
Nancy Salmon, Manager; Michael Murphy, Assistant. Frances Benton, Connie Burton, Phil Geraci, Donna Hartwell, Ruth Mustor, Scott Philbrick, Anne Rocchio, Dianne Ritson, Deborah Stone, Theresa Verville, Laura Woerner, Karen Wozmak; Ad Coordinators: David Wozmak, Mary Seaver; Advertising Production: Jane Preston, Steve Baldwin, Fiona Davies, Bruce Hedin
PHOTOGRAPHY
Sandra Dukette, Laurie Jennison, Irene Vail, Thomas Villeneuve, Robert M. Villeneuve
TYPESETTING
Marie Barker, Melody Bedell, Sara Bedell, Michele DesRochers, Jennifer Fay, Lynn Haines, Linda Locke, Debbie Nutting, Karen Stewart, Susan Weller
DESIGN
Denzel Dyer, Howard Happ, Dion LeGros, Laurie MacMillan, Joyce Pillarella, Susan Stevens, Donna Wohlfarth

The left bracket, [, replaces the up arrow used by Radio Shack to indicate exponentiation on our printouts. When entering programs published in *80 Micro*, you should make this change.

80 formats its program listings to run 64-characters wide, the way they look on your video screen. This accounts for the occasional wrap-around you will notice in our program listings. Don't let it throw you, particularly when entering assembly listings.

Article submissions from our readers are welcomed and encouraged. Inquiries should be addressed to: Submissions Editor, 80 Pine Street, Peterborough, NH 03458. Include an SASE for a copy of our writers' guidelines. Payment for accepted articles is made at a rate of approximately $50 per printed page; all rights are purchased. Authors of reviews should contact the Review Editor, 80 Pine Street, Peterborough, NH 03458.

PROOF NOTES

The editors look at the issues

This anniversary issue was coordinated by Michael Nadeau.

MANAGING EDITOR
Eric Maloney
SENIOR COPY EDITOR
Michael Nadeau
NEWS EDITOR
John P. Mello Jr.
REVIEW EDITOR
Janet Fiderio
NEW PRODUCTS EDITOR
Eric Grevstad
EDITORS
Lynn Rognsvoog, Kerry Leichtman, Caron Taylor, Deborah Sargent, Mary Ruth
TECHNICAL CONSULTANT
Jake Commander
TECHNICAL EDITORS
G. Michael Vose (Features, Editor)
Art Huston (Editor)
PRODUCTION EDITOR
Susan Gross
LAYOUT EDITORS
Joan Ahern, Bob Dukette, Sue Hays, Anne Vadeboncoeur
PROOFREADERS
Peter Bjornsen, Vinoy Laughner, Peg LePage, Louis Marini
EDITORIAL ADMINISTRATIVE
Carole Maccioci, Nancy Noyd

Subscriptions:
Problems with Subscriptions: Send a description of the problem and your current and/or most recent address to: *80 Micro*, Subscription Department, P.O. Box 981, Farmingdale, NY 11737.
Change of Address: Send old label or copy of old address and new address to: *80 Micro*, P.O. Box 981, Farmingdale, NY 11737. Please give eight weeks advance notice.
Microfilm: This publication is available in microform from University Microfilms International. United States address: 300 North Zeeb Road, Dept. P.R., Ann Arbor, MI 48106. Foreign address: 18 Bedford Row, Dept. P.R., London, WC1R4EJ, England.
Dealers: Contact Ginnie Boudrieau, Bulk Sales Manager, *80 Micro*, Pine St., Peterborough, NH 03458. (800) 343-0728.

There's no end in sight

Here we are, three years later and 400 pages bigger. It looks like *80 Micro* is a success. This Third-Anniversary Issue celebrates that success.

Regular readers can expect the same great utilities, programming tips, tutorials, games, and construction projects they get in every issue. First-time readers are in for a treat.

If you just bought your TRS-80, you'll find valuable articles on topics such as shopping for a printer, choosing a word processor, how a disk stores information, care and feeding of cassette systems, and why a voltage suppressor is a good idea.

We also asked David Lien, who was one of the people responsible for developing the TRS-80, to write a short piece for us. He talks about the early days and how the industry is beginning to realize the need for user-understandable documentation. We hope you find it interesting.

Oh yes—our surprise. When we were sitting around last spring trying to think of ideas for this Anniversary Issue, someone suggested writing a stereoscopic 3-D program for the Color Computer and binding in the 3-D glasses. Of course, we all laughed... and then looked at each other and said, "Why not?" We think you'll find the technique interesting enough to try it yourself.

In This Issue

Though most of the articles were from our files, we solicited several especially for this Anniversary Issue. These include "NODOS 80" by Thomas L. Quindry, the piece on shopping for a printer by Jim Hansen, Mike Vose's article on voltage suppressors, and, of course, the two 3-D pieces by Dennis Kitsz and Jake Commander.

NODOS 80 seemed like a good idea: Put a few of the better disk-based utilities published in past issues of *80 Micro* into one program for cassette users. Some of the utilities Tom included are not found in many commercial DOSes. Owners of cassette-based systems should find this article a real prize.

Jim Hansen is an engineer who designs printers for microcomputers, so when he tells you what to look for in a printer, you know he knows what he's talking about. After reading his advice, you'll feel confident when dealing with printer salesmen and reassured that you'll make a good choice.

The need for voltage suppressors is often overlooked by micro users. It takes only one good electrical surge to zap a disk, or only one lightning strike to burn out your computer. This inexpensive peripheral could be one of the best investments you'll make.

When we asked Dennis Kitsz to write a stereoscopic 3-D program for the Color Computer, his eyes lit up. Dennis has long touted the Color Computer as a technical marvel, and he jumped at the chance to do something new with it. Even if you don't own a Color Computer, you will find his explanation of how 3-D works interesting. In fact, we've published two sets of photos that you can see in 3-D even without using the special glasses.

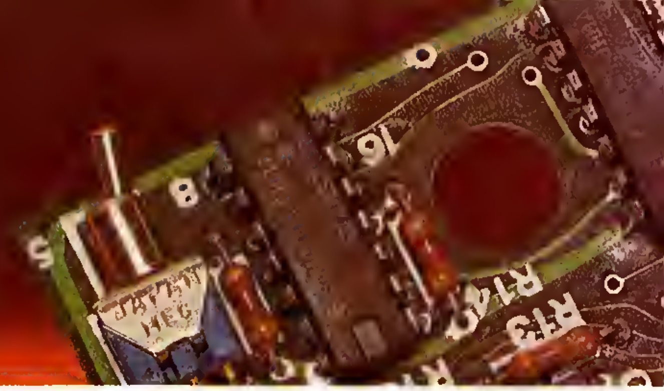
Jake Commander, our own technical wiz, was also eager to try his hand at 3-D. But Jake, having a reputation to uphold, had to break new ground. Even without the 3-D, Jake's program is interesting for its rotating-cube graphics techniques. It turned out to be more work than he expected, but you should find his efforts worthwhile.

The Future

Someday, *80 Micro*'s growth will level off—it's inevitable. But at the moment, there's no end in sight.

Not only will we be publishing more articles each month, but the quality of those articles will be better. We've been impressed with the submissions of late, and to give you an idea of what to expect in the near future, here is a sample

THE SWITCH



SWITCH TO 5"/8" DOUBLE DENSITY

LNDoubler 5/8

FEATURES

- 5- and 8-inch* disk drives
- Single- & double-density
- Any size and density in any mix
- Read Model I, II* and III disks
- 5- or 8-inch* system disk
- Single & double sided disk drives
- DOS+ 3.3.9 included, with Disk BASIC.
- 6 month warranty
- Up to 3.75 megabytes online
- Easy installation – plug-in & run
- Analog phase lock loop data separation
- Precision write precompensation
- Regulated power supply
- Guaranteed operation at 4MHz
- All contacts gold plated
- Solder masked & silk screened
- Runs under DOS+ 3.3.9, TRSDOS 2.3, NEWDOS 2.1, NEWDOS/80 1.0, LDOS, NEWDOS/80 2.0, and ULTRADOS
- Reads 40- and 35-track disks on 80-track drives
- FD1791 controller + your FD1771
- Fits Model I expansion interfaces
- Fits LNW expansion interfaces
- Track configurations to 80-tracks
- 5 inch disk storage increased to:
  161,280 bytes – 35-track SS/DD
  322,560 bytes – 35-track DS/DD
  184,320 bytes – 40-track SS/DD
  368,640 bytes – 40-track DS/DD
  368,640 bytes – 80-track SS/DD
  737,280 bytes – 80-track DS/DD
- 8 inch disk storage increased to:
  591,360 bytes – 77-track SS/DD
  1,182,720 bytes – 77-track DS/DD

SS: single-sided DS: double-sided
SD: single-density DD: double-density

COMPLETE – The *LNDoubler 5/8*, switches your Model I or LNW-80 into the most versatile computer you can own. The LNDoubler's switch allows you to boot from 5- or 8-inch system disks, and it's accessible from outside the interface. The *LNDoubler 5/8* comes with a double-density disk operating system (DOS+ 3.3.9), complete with BASIC and utility programs . . . ready to run your software NOW!

VERSATILE – Whether you want single-sided, double-sided, single- or double-density, 5- or 8-inch operation, complete versatility is here today! Any combination of 5- and 8-inch disk storage is possible with the *LNDoubler 5/8*. Each of your present 40-track, single-sided 5-inch drives will store up to 184,320 bytes (formatted storage) – that's an 80% increase in storage capacity for only half the cost of just one disk drive. With three 8-inch double-density, double-sided drives your Model I will have 3.75 Megabytes of online storage – that's more storage than a Model II or Model III!

ADVANCED – The *LNDoubler 5/8* is the most technically advanced, tested and reliable double-density board you can buy. The *LNDoubler 5/8* has more features, more options and more software support than any other product of its kind.

EASY TO INSTALL – The *LNDoubler 5/8* is easy to install. There are no traces to cut, no wiring to do, just a screwdriver and a few minutes of your time is all that is required. The instructions are fully illustrated for all interfaces. In minutes you will be 'up-and-running', and enjoying your computer as never before.

COMPARE – Compare features, compare quality, compare value, and make the **SWITCH** today!

Immediate delivery from stock – at your dealer **NOW** for only

$219.95

DEALERS – You too can make The Switch.

LNW RESEARCH CORPORATION

2620 WALNUT Tustin, CA. 92680 (714) 544-5744 (714) 641-8850

*8" drive operation requires special cable, 8" double-density requires 3.55MHz CPU speed-up modification or LNW-80 4MHz computer.

The 80 Micro *staff gets ready for 3-D.*

of articles we've recently bought:

- A build-your-own hi-res color monitor for the Model III for $150.
- A $5 CP/M mod.
- A great tutorial on the FLEX operating system for the Color Computer from the man who adapted it from the IBM system.
- A new column by Hardin Brothers on using Assembly-language subroutines in Basic programs.
- A series on using a micro in cryptanalysis by Karl Andreassen, the first of which appears in this Anniversary Issue.
- How to copyright and market your software.
- Many well-written tutorials on Basic programming that will appeal to the beginning to intermediate programmer.

We'll also be publishing timely reviews written by users and a new-product listing. And your favorite columns will continue: Feedback Loop, a question-and-answer forum; MONEY DOS a financial-investment advisor; Medical Opinion, advice for MDs; and 80 Applications, a hardware hacker's haven by Dennis Kitsz.

We think we've put together a pretty good collection of articles in this Anniversary Issue; there should be enough of everything to satisfy everyone. And if there isn't, let us know and we'll try to include it in the next one. ■

META TECHNOLOGIES

participating organization point marketing services

26111 Brush Avenue, Euclid Ohio 44132
CALL TOLL FREE 1-800-321-3552 TO ORDER
IN OHIO, call (216) 289-7500 (COLLECT)

PARAGON Gold magnetics™ DISKETTES

MTC's premium Single-Sided, Soft-Sectored DOUBLE-DENSITY, 5¼-inch diskettes with reinforcing HUB-RINGS. Individually 100% ERROR-FREE certified. Invest in GOLD!

PARAGON MAGNETICS GOLD $24.95

Scotch®

Soft-Sectored Diskettes

5-1/4" 1S/SDen (744-0)	$28.95
5-1/4" 1S/DDen (744D-0)	$31.95
5-1/4" 2S/DDen (745-0)	$39.95
8" 1S/SDen (740-0)	$29.95
8" 1S/DDen (741-0)	$37.95
8" 2S/DDen (743-0)	$49.95

Hard-Sectored Diskettes

5-1/4" 1S/DDen 10-sector (744-10)	$28.95
5-1/4" 1S/DDen 10-sector (744D-10)	$32.95
5-1/4" 2S/DDen 10-sector (745-10)	$39.95
5-1/4" 1S/SDen 16-sector (744-16)	$28.95
5-1/4" 1S/DDen 16-sector (744D-16)	$32.95
5-1/4" 2S/DDen 16-sector (745-16)	$39.95
8" 1S/SDen 32-sector (740-32)	$33.95
8" 1S/DDen 32-sector (741-32)	$39.95
8" 2S/DDen 32-sector (743-32)	$49.95

Supplies

5-1/4" Head cleaning kit	$29.95
8" Head cleaning kit	$29.95

Authorized Distributor Information Processing Products 3M

VERBATIM

Soft-Sectored Diskettes

5-1/4" 1S/DDen (MD525-01)	$26.95
5-1/4" 2S/DDen (MD550-01)	$39.95
5-1/4" 2S/4Den (MD557-01)	$51.50
8" 1S/DDen (FD34-8000)	$43.95

Hard-Sectored Diskettes

5-1/4" 1S/DDen 10-sector (MD525-10)	$26.95
5-1/4" 2S/DDen 10-sector (MD550-10)	$39.95
5-1/4" 2S/4Den 10-sector (MD557-10)	$51.50

'RINGS' & THINGS

HUB RING KIT for 5¼" disks	$10.95
HUB RING KIT for 8" disks	$12.95
REFILLS (50 Hub Rings)	$ 5.95
5¼-inch diskette case	$3.50
8-inch diskette case	$3.95
5¼-inch File Box for 50 diskettes	$24.95
8-inch File Box for 50 diskettes	$29.95

FILE BOX

DISKETTE STORAGE SYSTEM

$17.95* with SCOREPAC™

Storing 50 to 60 diskettes, this durable, smoke-colored acrylic unit provides easy access through the use of index dividers and adjustable tabs. Unique lid design provides dust-free protection and doubles as a carrying handle.

Reg. $24.95. . . . .NOW $17.95

* With the purchase of ONE

PARAGON GOLD

SCOREPAC™

Isn't it time that you scored . . .

A Great Gift Giving Idea!!

SCOREPAC™
w/20 Paragon Gold Diskettes $46.95

MODEL I/III SOFTWARE ACCESSORIES

APPARAT'S NEWDOS/80-V2 + MTC QUE CARD™	$ 129.95
PRACTICAL PERIPHERALS MICROBUFFER™	$ 149.95

Michael Shrayer's

ELECTRIC PENCIL VERSION II

SPECIFY MODEL I OR III

Disk Version	$79.95
Tape Version	$69.95

APPLE® OWNERS

Your first mystery is here!

THE CUSTOM APPLE® & OTHER MYSTERIES $24.95
by Hofacker & Floegel

Make your Apple® Computer perform tasks you never thought possible. Interface to stepping motors, fans, drive motors, etc. Remotely control and automatically monitor virtually any machine previously controlled by mechanical means. If you want to lower your heating costs or are just wondering how to hook that "new" printer to your Apple® you need this book! Volume 1 of IJG's Apple Information Series contains more than 180 pages of practical information and tested software.

"TRS-80™ DISK AND OTHER MYSTERIES"

by Harvard C. Pennington

TRS-80™ DISK $19.95

"OTHER MYSTERIES" VOLUME II

by James Farvour

MICROSOFT™ BASIC DECODED $26.95

"OTHER MYSTERIES" VOLUME III

by Dennis Kitsz

THE CUSTOM TRS-80™ $28.95

"OTHER MYSTERIES" VOLUME IV

BASIC FASTER AND BETTER

BASIC FASTER & BETTER	$26.95
BFBLIB subroutines diskette	$19.95
BFBDEM demonstration diskette	$19.95

APPLE is a registered trademark of Apple Computer Inc. TRS 80 is a trademark of the Radio Shack Division of Tandy Corporation. DATALIFE is a trademark of VERBATIM. PLAIN JANE, PARAGON MAGNETICS are trademarks of MTC. ©1982 by Meta Technologies Corporation

MOST ORDERS SHIPPED WITHIN ONE BUSINESS DAY

24 Hr. Bulletin Board (216)289-8689

PRICES IN EFFECT THRU December 31, 1982
Prices, Specifications, and Offerings subject to change without notice
8212

WE ACCEPT
- VISA
- MASTER CHARGE
- CHECKS
- MONEY ORDERS
- C.O.D.

- Add $3.00 for shipping & handling.
- $2.00 EXTRA for U.S. Mail delivery.
- $3.00 EXTRA for C.O.D.
- Ohio residents add 6.5% sales tax.

A Look Back

Through Level II Basic, tutorial-style documentation, and persistence, the TRS-80 Model I was born a user-friendly computer.

Ed. note: Dr. David Lien is one of the people who made the TRS-80 possible. He continues to support the Radio Shack computers by publishing books on Basic programming on the TRS-80—several of which are considered standards. Following are Dr. Lien's reflections on his role in the development of the Model I and the need for quality documentation.

It was 1976—only 6 years ago. Fort Worth was hot and humid.

I sat in the shabby old Radio Shack headquarters building discussing the possibility of a "personal computer." President Lou Kornfeld was justifiably skeptical.

"If I gave one of these to my wife for Christmas, she'd think I was some sort of nut," he said, clutching a prototype of the original Model I. Worded that way, the project didn't seem to make much sense.

"What can I really use it for?" others asked.

My answer: "Its use is limited only by your imagination."

That answer sounds as inadequate today as it did then.

At that time there was only Tiny Basic. No Level II Basic. No VisiCalc. No Scripsit. No Profile. No nothin! Just a game of Blackjack on an audio cassette.

I was retained to create a critical new interface between the engineers and the customers—a *tutorial-style* user's manual. It was to be a complete correspondence course on how to run a computer—without the correspondence! No one had ever heard of a "user's manual" for a computer written for beginners, and no one (including me) was completely sure it could be done.

I promised Radio Shack only one thing—I would "*take away the fear.*" They got the Model I user's manual for Level I Basic. The rest is history.

At the Creation

Don French, Steve Leininger, and I sat around a large old oak table, located in an entry where everyone bumped into it. The project was now well under way and it had a problem.

Don was the marketing genius who conceived the TRS-80, and who sensed the need for the engineer-consumer interface. He had asked me to solve that problem. There was just one additional little problem. No one at Radio Shack seemed to know much about the Basic language. I did, and agreed to help.

We sat all day at that big round table. I went through an exhaustive list of Basic words and explained their capabilities. Only 4K of ROM was available, and we had to decide which words to include in a Basic interpreter. It seemed like an impossible job, but at the end of the day Level I Basic was a reality.

The Breakthrough

The complete story of the TRS-80 has never been told. It is fascinating, but perhaps of interest mostly to historians.

From my perspective, the important breakthrough was recognition of the need for *premium*-quality "documentation" to accompany a highly technical product destined for a consumer market. Most important, that need was recognized and acted upon before the product was even designed. Radio Shack and Epson both had such vision. Count their profits!

It's amazing how few hardware and software manufacturers appreciate the importance of communicating with we end users in "end-user talk." Fewer still actually do it without insulting our intelligence. How many failures of good products can be attributed to not understanding the marketing value of correctly interfacing the product with the end user?

Let Engineering Do It?

"Documentation," "manuals," "books," or "instructions," by any name are usually an afterthought, remembered only when the product finally works and engineering turns it over to marketing. Marketing realizes that something has to be stuffed in the shipping box along with the widget, so they tell engineering, "You guys know more about this thing than anyone else, so you write the instructions." Engineering gathers up notes scribbled on the back of used computer printouts and does what it hates most—and does worst. The results are usually awful!

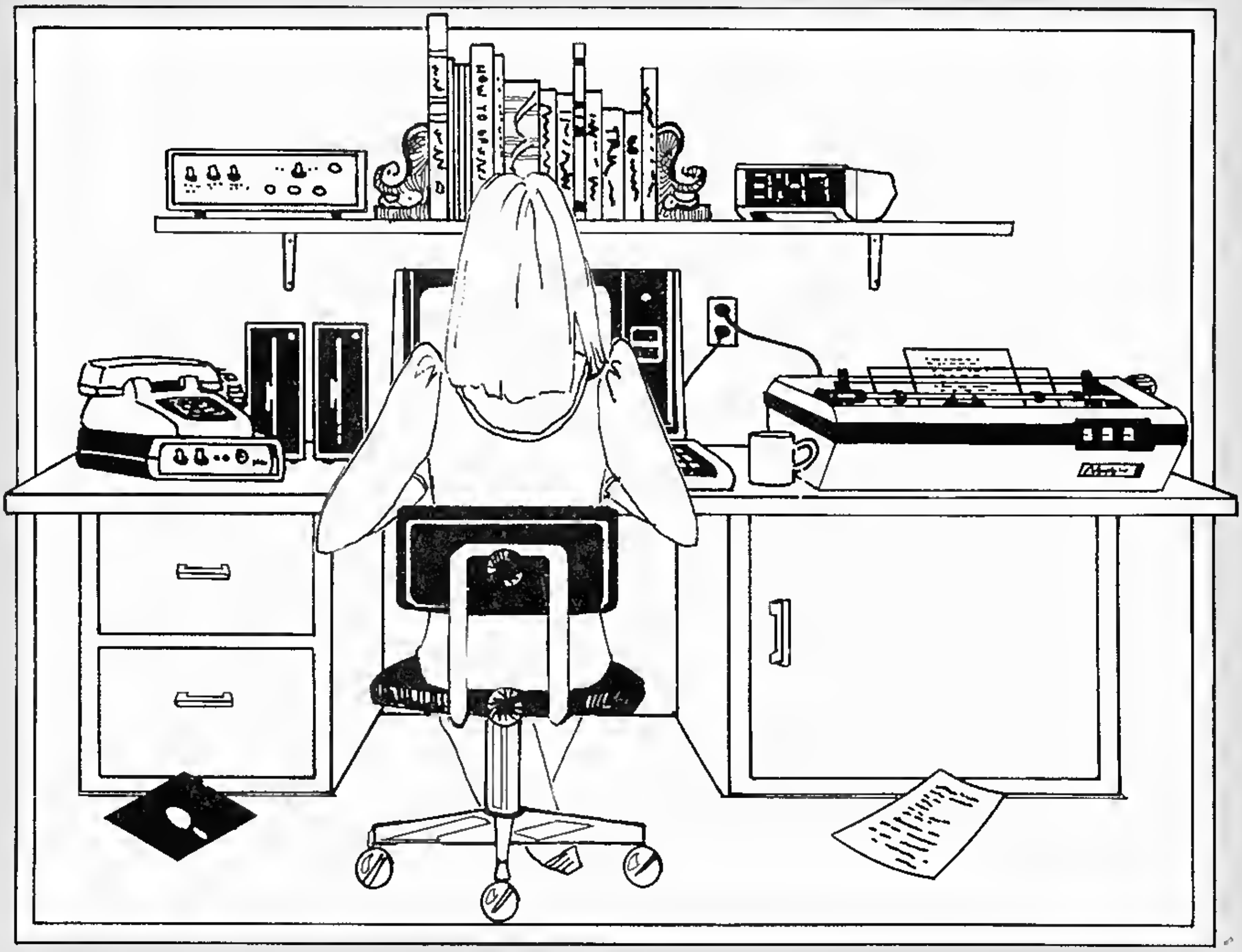
Since the products are usually well engineered, what can we do to get our money's worth? Not many are sold in sufficient volume to attract good supplemental documentation written by noncompany authors.

Friends, user groups, and local computer stores can sometimes help, but the best information source usually comes each month in the mail. Magazines like *80 Micro* with their specific product emphasis promote the free exchange of information and ideas, encouraging experimentation and learning. If you can't understand the manual, try a good magazine.

Specific articles like those that follow in this special Anniversary Edition help bridge the gap between the "official documentation" and *your* imagination.

Dr. David A. Lien
San Diego—1982

Part I. KNOW YOUR SYSTEM



Assembly Language Primer

by Bill Barden, Jr.

Bored by Basic, but unsure of taking the plunge to Assembly language? Bill Barden may give you just the encouragement you need.

Want a convincing demonstration that Assembly language is better than double density on the Model I, a 68000 in the Model II, a hard disk for the Model III, or lowercase for the Color Computer? Enter Program Listing 1 for your Model I/III (protect memory from &H7F00 on) or Program Listing 2 for your CoCo (CLEAR 200,&H3EFF) and run it.

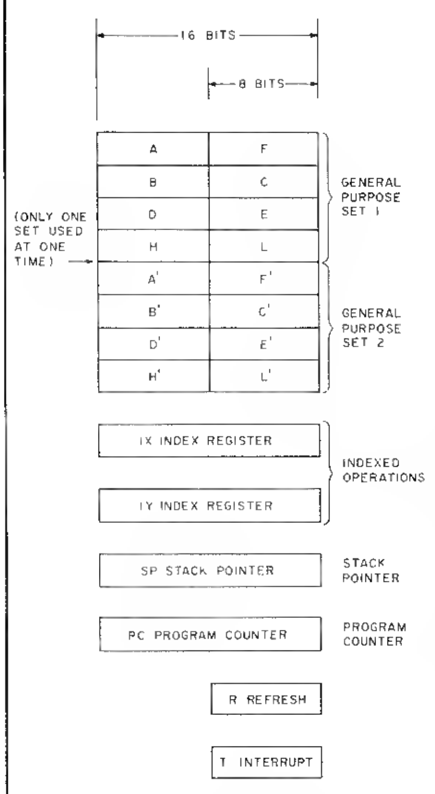


Fig. 1. Z80 CPU Registers

Each program is a simple bubble sort of random characters on the first half of your video screen. You'll see the characters rearrange themselves in alphabetical order. Time how long it takes to sort the data. Now enter the version shown in Program Listing 3 or 4 and run it. Time it. Incredible, eh? The first version took about 20 minutes while the second version took about 10 seconds!

As you might have guessed, the second version is in Assembly language. Assembly language is the difference between stuffing a TV chicken dinner into the microwave and preparing Coq au Vin from scratch. The results are similar, but one is convenient and adequate while the other is handcrafted and a gourmet delight.

In this article we'll look at what Assembly language is, how it differs from Basic and other higher-level languages, the advantages and disadvantages of Assembly language, assembler programs, and finally, give you a bibliography of materials to help you learn Assembly language.

Which Came First, Basic or Assembly Language?

Basic has special commands related to useful functions, such as addition, multiplication, exponentiation, processing of alphabetic text strings, printing and display of results, and arrays of data. The Basic commands are meant to be general-purpose functions that when strung together in a program accomplish useful results.

It is virtually impossible to build an electronic device that implements these commands directly. It's hard to conceive of being able to build a machine that would perform such diverse functions as finding the nth left characters of a string (LEFT$), printing a line of numeric and alphabetic characters in special format, and finding the SIN of an angle. There are just too many functions and in too many combinations to be able to build such a computer.

How do we implement Basic or other applications problems, such as spread sheets, word processing, or control of the Acme No-Cal Cheesecake production line? It's done on a microprocessor, a machine that performs rudimentary arithmetic and control functions. These low-level functions, when properly combined in a program, can implement the Basic commands, word processing searches, and rejection of cheesecakes that contain too many calories.

As a matter of fact, we've been looking at the whole topic in reverse. First came machine and Assembly language, and then came Basic. Computers were originally developed from the need to rapidly perform simple operations such as addition, subtraction, multiplication, and division. The first computers were high-speed calculators that evolved into machines capable of performing other functions, such as making decisions and comparisons. The current evolutionary state of computing machines uses Basic and applications programs.

Instruction Sets

Let's get down to the actual instruc-

tions that microprocessors can perform. There aren't very many. We'll look at the Z80 microprocessor used in the Model I, II, and III, and the 6809E microprocessor used in the CoCo. In spite of chauvinists, these microprocessors are roughly the same in power and capability.

Addition and Subtraction: Both the Z80 and 6809E have a number of instructions to add and subtract numbers, as you might expect. However, only smaller numbers can be processed—numbers that are 16 binary digits (bits) in length, maximum. These are binary numbers that are 0000000000000000-1111111111111111, representing decimal equivalents of 0–65,535, or (in another format) −32,768 to +32,767.

Any size of numbers *can* be processed by combining multiple adds and subtracts in a sequence of instructions, but the Z80 and 6809E instructions only operate on 8 or 16-bit numbers.

Multiplies and Divides: The Z80 has no multiplication or divide instructions; the 6809E has a multiply only. Believe it or not, microprocessor instructions are that rudimentary! Multiplies and divides must be done by stringing together adds and subtracts, along with other instructions, in a 20 or 30-instruction program to implement multiplies and divides.

Decision-Making Instructions: The Z80 and 6809E have instructions to make decisions. These instructions alter the flow of the program and are used continually. These *conditional jump* or *conditional branch* instructions jump to a new sequence of instructions *if* a condition is met. The condition tested is usually the result of some previous add, subtract, or comparison, and represents conditions such as *zero result*, *negative result*, *positive result*, *not equal*, *equal*, and others.

An *unconditional jump* also alters the sequence of instructions to be executed, and is always (unconditionally) done.

Logical and Bit-Manipulation Instructions: Basic generally operates on entire numeric quantities. However, the instructions of the Z80 and 6809E provide a *bit-manipulation* capability. ANDs and ORs provide logical functions on a binary-digit level, and *shifts* provide the ability to realign data on a bit basis. These instruction types are important to the microprocessor as they allow multiplies and divides to be implemented in programs, permit efficient storage of data, and provide control of input/output devices.

Subroutine Instructions: Subroutines are sets of instructions, ranging from several to thousands of instructions, that perform certain functions, ranging from comparing two numbers to sorting data into an alphabetized list. Rather than repeating the instructions every time they are required, they are stored in one set of locations in memory and *called* by different parts of the program. This saves a great deal of memory space. Both the Z80 and 6809E have instructions that allow subroutines to be called and for returns to the calling point.

Memory-Reference Instructions: Assembly-language programs are stored in memory (ROM or RAM), along with operands for the program. The operands are temporary results or constants for the program. The Z80 and 6809E have a large number of *memory-reference* instructions that pass 8 and 16-bit operands between memory and *CPU registers*, which are high-speed memory locations located within the Z80 or 6809E CPU (central-processing unit) itself. We'll look at the CPU registers shortly.

Input/Output Instructions: The Z80 has a few important instructions that allow 8 bits of data to be passed between the microprocessor CPU and the outside world—input/output devices such as printers, disk drives, and cassette recorders or their *I/O controllers*. (The 6809E handles I/O devices by *memory mapping* them as locations in memory.)

Chip Architecture

Before we look at the detailed instruction sets of the Z80 and 6809E, let's first look at the *architecture* of the microprocessors. This is just a fancy word for defining the internal structure of the microprocessor.

Each microprocessor contains tens of thousands of transistor equivalents. Many of these are related to the timing and control of the microprocessor, along with interpretation of the instruction set. The most important parts of the microprocessors from an Assembly-language standpoint, however, are the CPU registers, shown in Figs. 1 and 2.

The CPU registers are memory locations, but fast-access memory locations located within the microprocessor chip itself, and not addressed as part of external ROM or RAM. For this reason, they are called *registers*.

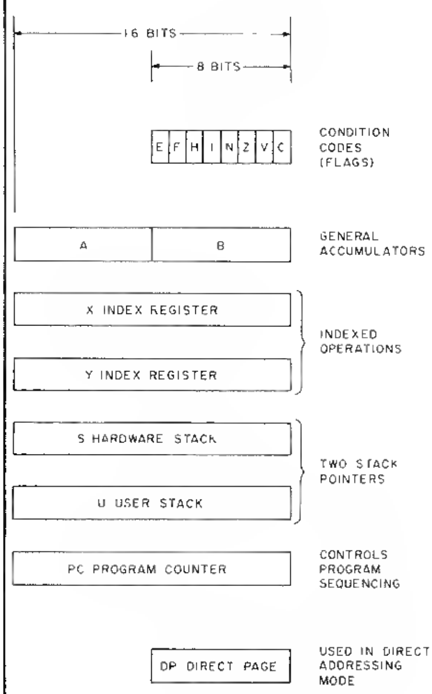


Fig. 2. 6809E CPU Registers

```
100 ' BUBBLE SORT DEMO FOR MODEL I/III
110 CLS:P=0
120 FOR I=15360 TO 15615
130 A=RND(127):IF A<32 THEN GOTO 130
140 POKE I,A
150 NEXT I
160 S=0
170 FOR I=15360 TO 15614
180 A=PEEK(I):B=PEEK(I+1):IF A<=B THEN GOTO 200
190 S=S+1:POKEI,B:POKE I+1,A
200 NEXT I
210 P=P+1:PRINT @ 530,"# SWAPS=";S;"PASS=";P
220 IF S<>0 THEN GOTO 160 ELSE GOTO 220
```

Program Listing 1. Basic Bubble Sort for Model I/III

```
100 ' BUBBLE SORT DEMO FOR COCO
110 CLS:P=0
120 FOR I=1024 TO 1024+255
130 A=RND(127):IF A<64 THEN GOTO 130
140 POKE I,A
150 NEXT I
160 S=0
170 FOR I=1024 TO 1278
180 A=PEEK(I):B=PEEK(I+1):IF A<=B THEN GOTO 200
190 S=S+1:POKEI,B:POKE I+1,A
200 NEXT I
210 P=P+1:PRINT @ 260,"# SWAPS=";S;"PASS=";P
220 IF S<>0 THEN GOTO 160 ELSE GOTO 220
```

Program Listing 2. Basic Bubble Sort for CoCo

Some of the registers hold intermediate results of arithmetic and other processing operations. Other registers are devoted to program control—the proper sequencing of instructions.

The A and B registers in the 6809E are general *accumulator* registers that hold temporary results. The Z80 A, B, C, D, E, H, and L registers (two sets) are the general-purpose accumulators in that chip.

Both microprocessors have *index registers* that allow the program to easily access sequential data. Both also have *stack pointers* that control access to a special part of external memory called the *stack*, which is used to hold the return points for subroutine calls, and temporary results.

Both chips also have a set of eight *flags*, collectively called the F register in the Z80 and the CC (condition codes) in the 6809E. Condition codes is probably a better term, because this special-purpose storage holds the results of adds, subtracts, comparisons, and other operations that can be tested for conditional branching.

The main program sequence register in both microprocessors is called the PC, or program counter. The PC points to the current *byte* of the program instruction. The instructions in the microprocessors are generally 1–4 bytes in length. A 400-instruction program, for example, would probably require about 800 bytes of RAM memory storage.

The PC register points to the next instruction in sequence. Branches change the contents of the program counter to allow different sets of instructions to be executed.

Data, or *operands*, can be interspersed with program instructions, or

Table 1. Z80 Instruction Set

ADC HL,ss	HL+ss+CY to HL	EX DE,HL	Exchange DE and HL
ADC A,r	A+r+CY to A	EXX	Set prime B-L active
ADC A,n	A+n+CY to A	HALT	Halt
ADC A,(HL)	A+(HL)+CY to A	IM 0	Set interrupt mode 0
ADC A,(IX+d)	A+(IX+d)+CY to A	IM 1	Set interrupt mode 1
ADC A,(IY+d)	A+(IY+d)+CY to A	IM 2	Set interrupt mode 2
ADD A,n	A+n to A	IN A,(n)	Load A with input from n
ADD A,r	A+r to A	IN r,(C)	Load r with input from (C)
ADD A,(HL)	A+(HL) to A	INC r	Increment r by one
ADD A,(IX+d)	A+(IX+d) to A	INC (HL)	Increment (HL) by one
ADD A,(IY+d)	A+(IY+d) to A	INC (IX+d)	Increment (IX+d) by one
ADD HL,ss	HL+ss to HL	INC (IY+d)	Increment (IY+d) by one
ADD IX,pp	IX+pp to IX	INC IX	Increment IX by one
ADD IY,rr	IY+rr to IY	INC IY	Increment IY by one
AND r	A AND r to A	INC ss	Increment register pair
AND n	A AND n to A	IND	Block I/O input from (C)
AND (HL)	A AND (HL) to A	INDR	Block I/O input, repeat
AND (IX+d)	A AND (IX+d) to A	INI	Block I/O input from (C)
AND (IY+d)	A AND (IY+d) to A	INIR	Block I/O input, repeat
BIT b,r	Test bit b of r	JP (HL)	Unconditional jump to (HL)
BIT b,(HL)	Test bit b of (HL)	JP (IX)	Unconditional jump to (IX)
BIT b,(IX+d)	Test bit b of (IX+d)	JP (IY)	Unconditional jump to (IY)
BIT b,(IY+d)	Test bit b of (IY+d)	JP cc,nn	Jump to nn if cc
CALL cc,nn	CALL subroutine at nn if cc	JP nn	Unconditional jump to nn
CALL nn	Unconditionally CALL nn	JR C,e	Jump relative if carry
CCF	Complement carry flag	JR e	Unconditional jump relative
CP r	Compare A:r	JR NC,e	Jump relative if no carry
CP n	Compare A:n	JR NZ,e	Jump relative if non-zero
CP (HL)	Compare A:(HL)	JR Z,e	Jump relative if zero
CP (IX+d)	Compare A:(IX+d)	LD A,(BC)	Load A with (BC)
CP (IY+d)	Compare A:(IY+d)	LD A,(DE)	Load A with (DE)
CPD	Block Compare, no repeat	LD A,I	Load A with I
CPDR	Block Compare, repeat	LD A,(nn)	Load A with location nn
CPI	Block Compare, no repeat	LD A,R	Load A with R
CPIR	Block Compare, repeat	LD (BC),A	Store A to (BC)
CPL	Complement A (one's complement)	LD (DE),A	Store A to (DE)
DAA	Decimal Adjust A	LD (HL),n	Store n to (HL)
DEC r	Decrement r by one	LD dd,nn	Load register pair with nn
DEC (HL)	Decrement (HL) by one	LD dd,(nn)	Load register pair with location nn
DEC (IX+d)	Decrement (IX+d) by one	LD HL,(nn)	Load HL with location nn
DEC (IY+d)	Decrement (IY+d) by one	LD (HL),r	Store r to (HL)
DEC IX	Decrement IX by one	LD I,A	Load I with A
DEC IY	Decrement IY by one	LD IX,(nn)	Load IX with nn
DEC ss	Decrement register pair	LD IX,nn	Load IX with location nn
DI	Disable interrupts	LD (IX+d),n	Store n to (IX+d)
DJNZ e	Decrement B and JR if B≠0	LD (IX+d),r	Store r to (IX+d)
EI	Enable Interrupts	LD IY,nn	Load IY with nn
EX (SP),HL	Exchange (SP) and HL	LD IY,(nn)	Load IY with location nn
EX (SP),IX	Exchange (SP) and IX	LD (IY+d),n	Store n to (IY+d)
EX (SP),IY	Exchange (SP) and IY		
EX AF, AF'	Set prime AF active		

Table 1 continues

can be kept in a separate area. The program itself must be set up to *jump/branch* over the data areas and to keep the flow of instructions in proper order.

Generally, the instruction steps are located in sequential order. Instruction execution consists of two parts, the *fetch cycle*, which fetches the 1, 2, 3, or 4 bytes of the instruction from memory and decodes it into the proper instruction, and the *execute cycle*, which may access memory again in different locations to read in or write out memory operands. This process is

```
100 ' BUBBLE SORT DEMO WITH AL SORT
101 DATA 14,0,221,33,0,60,6,254,221,126,0
102 DATA 221,86,1,186,40,10,56,8,14,1,221,114,0
103 DATA 221,119,1,221,35,16,233,203,65,32
104 DATA 221,201
105 FOR I=32512 TO 32547
106 READ A
107 POKE I,A
108 NEXT I
109 DEFUSR0=32512
110 CLS
120 FOR I=15360 TO 15615
130 A=RND(127):IF A<32 THEN GOTO 130
140 POKE I,A
150 NEXT I
160 A=USR0(0)
170 GOTO 170
```

Program Listing 3. Basic/Assembly-language Bubble Sort for Model I/III

Table 1 continued

LD (IY+d),r	Store r to (IY+d)
LD (nn),A	Store A to location nn
LD(nn),dd	Store register pair to location nn
LD (nn),HL	Store HL to location nn
LD (nn),IX	Store IX to location nn
LD (nn),IY	Store IY to location nn
LD R,A	Load R with A
LD r,r'	Load r with r'
LD r,n	Load r with n
LD r,(HL)	Load r with (HL)
LD r,(IX+d)	Load r with (IX+d)
LD r,(IY+d)	Load rf with (IY+d)
LD SP,HL	Load SP with HL
LD SP,IX	Load SP with IX
LD SP,IY	Load SP with IY
LDD	Block load, f'ward, no repeat
LDDR	Block load, f'ward, repeat
LDI	Block load, b'ward, no repeat
LDIR	Block load, b'ward, repeat
NEG	Negate A (two's complement)
NOP	No operation
OR r	A OR r to A
OR n	A OR n to A
OR (HL)	A OR (HL) to A
OR (IX+d)	A OR (IX+d) to A
OR (IY+d)	A OR (IY+d) to A
OTDR	Block output, b'ward, repeat
OTIR	Block output, f'ward, repeat
OUT (C),r	Output r to (C)
OUT (n),A	Output A to port n
OUTD	Block output, b'ward, no rpt
OUTI	Block output, f'ward, no rpt
POP IX	Pop IX from stack
POP IY	Pop IY from stack
POP qq	Pop qq from stack
PUSH IX	Push IX onto stack
PUSH IY	Push IY onto stack
PUSH qq	Push qq onto stack
RES b,r	Reset bit b of r
RES b,(HL)	Reset bit b of (HL)
RES b,(IX+d)	Reset bit b of (IX+d)
RES b,(IY+d)	Reset bit b of (IY+d)
RET	Return from subroutine
RET cc	Return from subroutine if cc
RETI	Return from interrupt
RETN	Return from non-maskable int
RL r	Rotate left thru carry r
RL (HL)	Rotate left thru carry (HL)
RL (IX+d)	Rotate left thru carry (IY+d)
RL (IY+d)	Rotate left thru carry (IY+d)
RLA	Rotate A left thru carry
RLC r	Rotate left circular r
RLC (HL)	Rotate left circular (HL)
RLC (IX+d)	Rotate left circular (IX+d)
RLC (IY+d)	Rotate left circular (IY+d)
RLCA	Rotate left circular A
RLD	Rotate bcd digit left (HL)
RR r	Rotate right thru carry r
RR (HL)	Rotate right thru carry (HL)
RR (IX+d)	Rotate right thru cy (IX+d)
RR (IY+d)	Rotate left thru cy (IY+d)
RRA	Rotate A right thru carry
RRC r	Rotate r right circular
RRC (HL)	Rotate (HL) right circular
RRC (IX+d)	Rotate (IX+d) right circular
RRC (IY+d)	Rotate (IY+d) right circular
RRCA	Rotate A right circular
RRD	Rotate bcd digit right (HL)
RST p	Restart to location p
SBC A,r	A-r-CY to A
SBC A,n	A-n-CY to A
SBC A,(HL)	A-(HL)-CY to A
SBC A,(IX+d)	A-(IX+d)-CY to A
SBC A,(IY+d)	A-(IY+d)-CY to A
SBC HL,ss	HL-ss-CY to HL
SCF	Set carry flag
SET b,(HL)	Set bit b of (HL)
SET b,(IX+d)	Set bit b of (IX+d)
SET b,(IY+d)	Set bit b of (IY+d)
SET b,r	Set bit b of r
SLA r	Shift r left arithmetic
SLA (HL)	Shift (HL) left arithmetic
SLA (IX+d)	Shift (IX+d) left arithmetic
SLA (IY+d)	Shift (IY+d) left arithmetic
SRA r	Shift r right arithmetic
SRA (HL)	Shift (HL) right arithmetic
SRA (IX+d)	Shift (IX+d) right arithmetic
SRA (IY+d)	Shift (IY+d) right arithmetic
SRL r	Shift r right logical
SRL (HL)	Shift (HL) right logical
SRL (IX+d)	Shift (IX+d) right logical
SRL (IY+d)	Shift (IY+d) right logical
SUB r	A-r to A
SUB n	A-n to A
SUB (HL)	A-(HL) to A
SUB (IX+d)	A-(IX+d) to A
SUB (IY+d)	A-(IY+d) to A
XOR r	A EXCLUSIVE OR r to A
XOR n	A EXCLUSIVE OR n to A
XOR (HL)	A EXCLUSIVE OR (HL) to A
XOR (IX+d)	A EXCLUSIVE OR (IX+d) to A
XOR (IY+d)	A EXCLUSIVE OR (IY+d) to A

shown in Fig. 3.

Each instruction takes about 2-12 microseconds or more to be performed—that's 2-12 millionths of a second. Although very fast (in comparison to Basic's hundreds of statements per second), don't forget that we're doing very rudimentary operations, and that it sometimes takes a large number of these instructions in sequence to accomplish the same thing that can be done in one Basic statement.

Addressing Modes

We're getting to the instruction set. Before we do, though, we should discuss instruction addressing modes.

Some microprocessor instructions are very simple. The Z80 SCF instruction, for example, sets the Carry flag to a 1; the 6809E CLRA clears the A register to zeros.

Other instructions are more complex. About the most complex Z80 instruction, the LDIR, moves a block of memory from one location to another; the 6809E MUL multiplies two numbers in the A and B registers and puts the product back into A and B.

To save memory space, the Z80 and 6809E have varying types of *addressing modes* for the instructions. In general, the instruction is made as short (as few bytes) as possible. However, the instruction length is a function of the instruction action.

An example: If an instruction adds a number from memory to the contents of a CPU register, it probably requires that the address of the memory operand must be in the instruction itself, and that takes 16 bits, or 2 bytes, as the operand may be located at any memory location from 0-65,535. Add another byte for an *operation code* to denote the type of instruction, and you have a 3-byte instruction. The STF and CLRA discussed above, though, can be made 1 byte long, as they require no memory address for an operand.

The Z80 and 6809E have a large number of addressing modes to provide efficient instruction storage and flexibility. Some of the modes are *implied*, where the instruction function is simple with no external operands; *immediate*, where the operand is part of the instruction itself (rather than holding a constant somewhere else in memory); *direct*, where the operand is addressed with a 2-byte operand address;

```
100 ' BUBBLE SORT DEMO FOR COCO WITH AL
101 DATA 16,142,0,0,142,4,0,166,132,230,1
102 DATA 161,1,35,8,16,142,0,1,231,132
103 DATA 167,1,48,1,140,4,255,38,233
104 DATA 16,140,0,0,38,220,57
105 FOR I=&H3F00 TO &H3F24
106 READ A: POKE I,A
107 NEXT I
108 DEFUSR0=&H3F00
110 CLS
120 FOR I=1024 TO 1024+255
130 A=RND(127):IF A<64 THEN GOTO 130
140 POKE I,A
150 NEXT I
160 A=USR0(0)
170 GOTO 170
```

Program Listing 4. Basic/Assembly-language Bubble Sort for CoCo

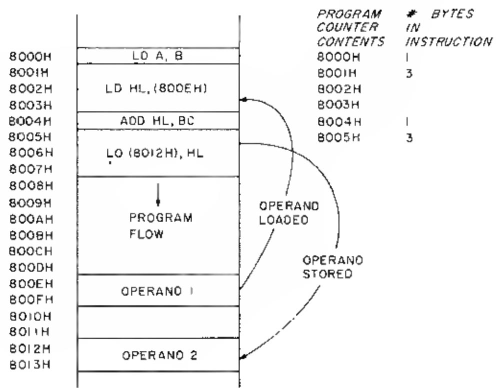


Fig. 3. Program Flow

WE ARBITRARILY STARTED AT 8000H

LOC	CONTENTS	INSTRUCTION	
8000	21-00-00	LD	HL,0
3	01-64-00	LD	BC,100
6	09	ADD	HL,BC
7	0B	DEC	BC
8	78	LD	A,B
9	B1	OR	C
A	C2-06-80	JP	NZ, $\boxed{8006}$
D	C9	RET	

THIS VALUE IS AN "ABSOLUTE" LOCATION

"OPCODE" IN FIRST BYTE

MAY BE 1-3 ADDITIONAL BYTES FOR ADDRESS OR OTHER OPERANDS

H = HEX#

Fig. 4. Hand Assembly, Z80

and *indexed*, where the index registers point to the operand or the starting location of sequential data.

Having a large number of addressing modes has one drawback: It makes it confusing for the beginner. He not only has to learn the *instruction set* of the microprocessor, he has to learn the addressing modes and for what instructions they are used, as well.

Instruction Sets

Well, we finally made it to the instruction sets of the Z80 and 6809E. The instruction sets for both microprocessors are typical of other microprocessors such as the 6800, 6502 (used in the Apple and Commodore), and 8088 (IBM PC). They are shown in Tables 1 and 2 in *mnemonic* form. The mnemonics are simply shorthand ways of representing the instructions—it's much easier to say, "LDA," than to say, "Load the contents of the A accumulator with an 8-bit memory location value."

Table 2. 6809E Instruction Set

8-BIT ACCUMULATOR AND MEMORY INSTRUCTIONS

		Addressing Modes								
Mnemonic(s)	**Operation**	**Implied**	**Immediate**	**Direct**	**Extended**	**Extended Indirect**	**Indexed**	**Indexed Indirect**	**Relative**	**Relative Indirect**
ADCA, ADCB	Add memory to accumulator with carry	—	x	x	x	x	x	x	x	x
ADDA, ADDB	Add memory to accumulator	—	x	x	x	x	x	x	x	x
ANDA, ANDB	And memory with accumulator	—	x	x	x	x	x	x	x	x
ASL	Arithmetic shift left memory location	—	—	x	x	x	x	x	x	x
ASLA, ASLB	Arithmetic shift left accumulator	x	—	—	—	—	—	—	—	—
ASR	Arithmetic shift right memory location	—	—	x	x	x	x	x	x	x
ASRA, ASRB	Arithmetic shift right accumulator	x	—	—	—	—	—	—	—	—
BITA, BITB	Bit test memory with accumulator	—	x	x	x	x	x	x	x	x
CLR	Clear memory location	—	—	x	x	x	x	x	x	x
CLRA, CLRB	Clear accumulator	x	—	—	—	—	—	—	—	—
CMPA, CMPB	Compare memory with accumulator	—	x	x	x	x	x	x	x	x
COM	Complement memory location	—	—	x	x	x	x	x	x	x
COMA, COMB	Complement accumulator	x	—	—	—	—	—	—	—	—
DAA	Decimal adjust A-accumulator	x	—	—	—	—	—	—	—	—
DEC	Decrement memory location	—	—	x	x	x	x	x	x	x
DECA, DECB	Decrement accumulator	x	—	—	—	—	—	—	—	—
EORA, EORB	Exclusive or memory with accumulator	—	x	x	x	x	x	x	x	x
EXG R1, R2	Exchange R1 with R2 (R1, R2=A, B, CC, DP)	x	—	—	—	—	—	—	—	—
INC	Increment memory location	—	—	x	x	x	x	x	x	x
INCA, INCB	Increment accumulator	x	—	—	—	—	—	—	—	—
LDA, LDB	Load accumulator from memory	—	x	x	x	x	x	x	x	x
LSL	Logical shift left memory location	—	—	x	x	x	x	x	x	x
LSLA, LSLB	Logical shift left accumulator	x	—	—	—	—	—	—	—	—
LSR	Logical shift right memory location	—	—	x	x	x	x	x	x	x
LSRA, LSRB	Logical shift right accumulator	x	—	—	—	—	—	—	—	—
MUL	Unsigned multiply (AXB→D)	x	—	—	—	—	—	—	—	—
NEG	Negate memory location	—	—	x	x	x	x	x	x	x
NEGA, NEGB	Negate accumulator	x	—	—	—	—	—	—	—	—
ORA, ORB	Or memory with accumulator	—	x	x	x	x	x	x	x	x
ROL	Rotate memory location left	—	—	x	x	x	x	x	x	x
ROLA, ROLB	Rotate accumulator left	x	—	—	—	—	—	—	—	—
ROR	Rotate memory location right	—	—	x	x	x	x	x	x	x
RORA, RORB	Rotate accumulator right	x	—	—	—	—	—	—	—	—
SBCA, SBCB	Subtract memory from accumulator with borrow	—	x	x	x	x	x	x	x	x
STA, STB	Store accumulator to memory	—	—	x	x	x	x	x	x	x
SUBA, SUBB	Subtract memory from accumulator	—	x	x	x	x	x	x	x	x
TST	Test memory location	—	—	x	x	x	x	x	x	x
TSTA, TSTB	Test accumulator	x	—	—	—	—	—	—	—	—
TFR,R1,R2	Transfer R1 to R2 (R1, R2=A, B, CC, DP)	x	—	—	—	—	—	—	—	—

NOTE: A and B may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 2 continues

Machine Language

We've been discussing the instruction sets (or instruction repertoire) of both the Z80 and 6809E, but haven't described how we enter them into the system. The most rudimentary way to do this is by *machine language*.

We could sit down and write a program to add the numbers 1–100 by using the instruction-set mnemonics from Table 1. We've done that in Figs. 4 and 5, for both the Z80 and 6809E.

The next step is to translate the mnemonics into the proper machine-language binary data values. This involves looking up the proper code from a table. The table values are given in hexadecimal, a shorthand form of binary. (Four binary digits equal one hexadecimal digit: 0000–1001 become hex 0–9, while 1010, 1011, 1100, 1101, 1110, and 1111 become hex A–F.)

The first byte in the instruction is usually an *opcode*, while remaining bytes are memory addresses or intermediate addressing values; some knowledge of the instruction length is required to properly *hand assemble* the code.

The hand-assembled code shown in Figs. 4 and 5 works and this is one way to generate the machine-language code. Early programmers had to do this for the very first machines. However, hand-assembly is very tedious, and it's extremely easy to make errors. Also, if more instructions are added, or if instructions are deleted, then the process has to be repeated for the entire program.

Why Not Let The Computer Do It?

Early in the computer game, programmers, being a shiftless lot at best, decided to let the computer do all the work. The result was an *assembler*, a computer program that automatically assembles the mnemonics into the proper machine-language code.

You can see how assemblers are implemented. The assembler looks at a mnemonic representing the opcode and at the operands for each instruction. It then looks up the proper machine-language code in a table and constructs an instruction of the proper length and operand addresses.

About the only subtlety in the assembly operation is the question of location *references*. In a hand assembly, the programmer must fill in the addresses of operands on a *second pass* after he assembles the skeleton of all the instructions and finds where in memory each instruction or constant is located. He can't put in the address of each operand or instruction beforehand, because he doesn't know how many bytes will be present before the operand or instruction appears. (If he wants to jump to a further point in the program, for example, he doesn't know what location that jump point will be at until after assembly.)

The assembler program solves the problem by letting the programmer refer to locations by symbols, rather than absolute memory locations. You might, for example, jump ahead to NEXT instead of location 8000 hexadecimal (32768 decimal in RAM).

This symbolic form of the two programs is shown in Listings 5 and 6, assembled this time by Z80 and 6809E assembler programs. Note that the machine-language code on the left side of the listing is largely identical to the hand assembly.

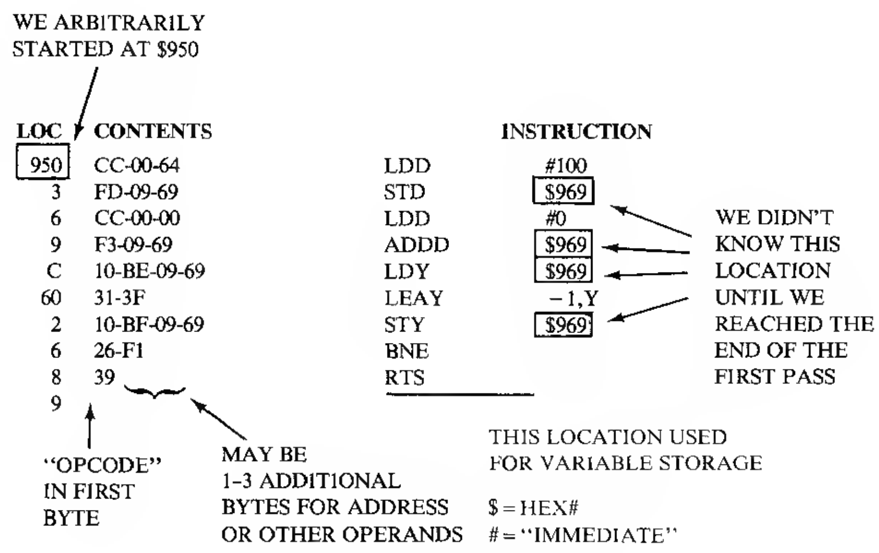


Fig. 5. Hand Assembly, 6809E

```
                  00100 ;***ADD THE NUMBERS FROM 1 TO 100 - MODEL I/III***
0000 210000       00110 ADDNUM  LD      HL,0          ;ZERO TO TOTAL
0003 016400       00120         LD      BC,100        ;INITIALIZE CURRENT #
0006 09           00130 ADD010  ADD     HL,BC           ;ADD IN CURRENT #
0007 0B           00140         DEC     BC              ;#-1
0008 78           00150         LD      A,B             ;TEST FOR 0
0009 B1           00160         OR      C
000A C20600       00170         JP      NZ,ADD010       ;GO IF NOT 0
000D C9           00180         RET                   ;RETURN
0000              00190         END
00000 Total errors
```

Program Listing 5. Symbolic Assembly for Z80

```
                        00100 ***ADD THE NUMBERS FROM 1 TO 100 - COCO***
0950 CC   0064          00110 ADDNUM  LDD     #100    ZERO TO A,B
0953 FD   0969          00120         STD     LOC     INITIALIZE CURRENT #
0956 CC   0000          00130         LDD     #0      INITIALIZE TOTAL
0959 F3   0969          00140 ADD010  ADDD    LOC       ADD IN CURRENT #
095C 10BE 0969          00150         LDY     LOC       GET CURRENT #
0960 31   3F            00160         LEAY    -1,Y      #-1
0962 10BF 0969          00170         STY     LOC       STORE
0966 26   F1            00180         BNE     ADD010    GO IF # NOT 0
0968 39                 00190         RTS             RETURN
0969                    00200 LOC     RMB     2       VARIABLE
          0000          00210         END
00000 TOTAL ERRORS

ADD010  0959
ADDNUM  0950
LOC     0969
```

Program Listing 6. Symbolic Assembly for 6809E

Assembly Language

Assembly language, then, is just a symbolic representation of machine language to make it possible for the programmer to feed in instructions to the assembler program for automatic assembly.

Program Listings 7 and 8 show typical Assembly-language listings for the bubble-sort programs discussed in the first part of this article. The portion on the right half is the actual symbolic Assembly language entered by the programmer; the left portion is the assembler-generated machine code, locations at which the code will go in memory, and edit line numbers. We've annotated the code heavily to give you an idea how the programs work.

The assembler translates the *source code* keyed in by the programmer into *object code*. The object code is very close to the machine-language code that the microprocessor uses, but contains other information for use in loading the program—such things as the load location, file name, constant data, and checksums. The object code is normally loaded from cassette or disk, and the source code can be written as source files on the same media.

```
                    00100 ;****BUBBLE SORT DEMO - MODEL I/I11 AL***
                    00110 ;SORTS ONE-CHARACTER ENTRIES IN FIRST 256
                    00120 ;BYTES OF SCREEN
0000 0E00           00130 BUBSRT  LD      C,0             ;C IS "CHANGE FLAG"
0002 DD21003C       00140         LD      IX,3C00H        ;START OF SCREEN
0006 06FE           00150         LD      B,254           ;FOR 256 ENTRIES-1
0008 DD7E00         00160 BUB010  LD      A,(1X)           :GET I ENTRY
000B DD5601         00170         LD      D,(1X+1)         :GET I+1 ENTRY
000E BA             00180         CP      D                ;COMPARE A TO D
000F 280A           00190         JR      Z,BUB020         :GO IF EQUAL
0011 3808           00200         JR      C,BUB020         :GO IF I<1+1
0013 0E01           00210         LD      C,1              ;SET SWAP FLAG
0015 DD7200         00220         LD      (IX),D           :SWAP ENTRIES
0018 DD7701         00230         LD      (IX+1),A
001B DD23           00240 BUB020  INC     IX               :SCREEN ADDRESS+1
001D 10E9           00250         DJNZ    BUB010           ;DO FOR 254 PAIRS
001F CB41           00260         B1T     0,C             ;TEST C=0 OR 1 FOR CHANGE
0021 20DD           00270         JR      NZ,BUBSRT       ;SWAP, ANOTHER PASS
0023 C9             00280         RET                     ;RETURN TO BASIC
0000                00290         END
00000 Total errors
```

Program Listing 7. Bubble-sort Demo for Models I/III

Table 2 continued

16-BIT ACCUMULATOR ANU MEMORY INSTRUCTIONS

Mnemonic(s)	Operation	Addressing Modes: Implied	Immediate	Direct	Extended	Extended Indirect	Indexed	Indexed Indirect	Relative	Relative Indirect
ADDD	Add memory to D accumulator	—	x	x	x	x	x	x	x	x
CMPD	Compare memory with D accumulator	—	x	x	x	x	x	x	x	x
EXG D, R	Exchange D with X, Y, S, U, or PC	x	—	—	—	—	—	—	—	—
LDD	Load D accumulator from memory	—	x	x	x	x	x	x	x	x
SEX	Sign Extend	x	—	—	—	—	—	—	—	—
STD	Store D accumulator to memory	—	—	x	x	x	x	x	x	x
SUBD	Subtract memory from D accumulator	—	x	x	x	x	x	x	x	x
TFR D,R	Transfer D to X. Y. S, U, or PC	x	—	—	—	—	—	—	—	—
TFR R, D	Transfer X. Y, S, U, or PC to D	x	—	—	—	—	—	—	—	—

INDEX REGISTER/STACK POINTER INSTRUCTIONS

Mnemonic(s)	Operation	Addressing Modes: Implied	Immediate	Direct	Extended	Extended Indirect	Indexed	Indexed Indirect	Relative	Relative Indirect
CMPS, CMPU	Compare memory with stack pointer	—	x	x	x	x	x	x	x	x
CMPX, CMPY	Compare memory with index register	—	x	x	x	x	x	x	x	x
EXG R1, R2	Exchange D,X,Y,S,U, or PC with D,X,Y,S,U, or PC	x	—	—	—	—	—	—	—	—
LEAS, LEAU	Load effective address into stack pointer	—	—	—	—	x	x	x	x	x
LEAX, LEAY	Load effective address into index register	—	—	—	—	x	x	x	x	x
LDS, LDU	Load stack pointer from memory	—	x	x	x	x	x	x	x	x
LDX, LDY	Load index register from memory	—	x	x	x	x	x	x	x	x
PSHS	Push any register(s) onto hardware stack (except S)	x	—	—	—	—	—	—	—	—
PSHU	Push any register(s) onto user stack (except U)	x	—	—	—	—	—	—	—	—
PULS	Pull any register(s) from hardware stack (except S)	x	—	—	—	—	—	—	—	—
PULU	Pull any register(s) from user stack (except U)	x	—	—	—	—	—	—	—	—
STS, STU	Store stack pointer to memory	—	—	x	x	x	x	x	x	x
STX, STY	Store index register to memory	—	—	x	x	x	x	x	x	x
TFR R1, R2	Transfer D, X, U, or PC to D, X, S, U, or PC	x	—	—	—	—	—	—	—	—
ABX	Add B-accumulator to X (unsigned)	x	—	—	—	—	—	—	—	—

Table 2 continues

Bells and Whistles on Assemblers

The process shown above is the most rudimentary form of automatic assembly by an assembler program. There are many more niceties that can be added to the basic assembler process.

Editing Capability: Almost all assemblers contain a built-in editor that can be used to create and modify source-code files from cassette or disk. The Radio Shack Series I Editor/Assembler and the Radio Shack

EDTASM+ editors, for example, have a built-in editor that works identically to the editor in Basic, allowing line-by-line or character editing. The Misosys EDAS Editor/Assembler has quite a bit more capability, with block moves and copies.

Pseudo-ops: All assemblers have *pseudo-operations*—assembler commands that are instructions to the assembler. The pseudo-op ORG, for example, sets the location of the program. EQU equates one symbol to another. DEFB, DEFW, FCB, and FCW generate constant data in either byte or word format. DEFM and FCC generate messages, or ASCII text data. DEFS and RMB reserve areas of memory, similar to allocating array storage in Basic. END ends the source-code statements.

The pseudo-ops described above are common for basic editor/assemblers such as the Series I for the Z80 and EDTASM+ for the CoCo.

In-Memory Assembly: We mentioned that object code could be stored on cassette and disk for later loading. This is the conventional approach carried over from olden days of computers 20 years ago. New assemblers, however, are much more interactive and allow you to load the object code directly into RAM memory for debugging. (Oh yes, most Assembly-language code will not run the first time, or the second time, or the....) After a debugging session, the editor/assembler can be re-entered for a new assembly to clean up errors.

The Misosys EDAS and Microsoft EDTASM+ provide this in-memory capability for the Model I/III, and the Radio Shack EDTASM+ for the CoCo also gives you this option. (The CoCo EDTASM+ is a functional subset of the Microsoft EDTASM+, which is probably the ultimate editor/assembler/debug package.)

Debug Capability: The Microsoft EDTASM+ for the I/III and the CoCo EDTASM+ also include a built-in debug package called ZBUG. ZBUG provides symbolic debugging, which allows you to examine and change memory locations by their assembly-time symbolic names. In addition you can *disassemble* existing machine-language code into equivalent instruction mnemonics, a handy feature to allow you to see how a program works if the source code is not available. Other ZBUG features are too numerous to mention, but suffice it to say you can do a great deal. After your debugging, a simple one-character entry brings you back to Edit/Assemble with the source code intact, without reloading!

Macro Capability: Now we're getting esoteric. Certain assemblers, such as the Microsoft EDTASM+ for the Model I/III, allow you to enter *macros*, which are somewhat like predefined subroutines, but at a source-code level. By a simple call with a macro name you can invoke assembly of several lines to hundreds of lines of source code. This differs from a subroutine in that the code exists in many places, but does save a great deal of Assembly-language coding.

Relocatable Object Code: Certain other sophisticated assemblers, such as the Radio Shack Disk Assembler for the Model II, use a somewhat different structure in generating object code. Each set of object code is generated as a *relocatable object module*. Loading the complete program involves loading several to dozens of these object modules together in a *link-load* process. The advantage of relocatable object modules is that each programmer may create his own object modules with common entry points and external variable storage defined by special pseudo-ops; at load time all entry points and variables will be linked together automatically. Another advantage is the process permits larger As-

Table 2 continued

BRANCH INSTRUCTIONS — Addressing Modes

Mnemonic(s)	Operation	Implied	Immediate	Direct	Extended	Extended Indirect	Indexed	Indexed Indirect	Relative	Relative Indirect
BCC, LBCC	Branch if carry clear	—	—	—	—	—	—	—	x	—
BCS, LBCS	Branch if carry set	—	—	—	—	—	—	—	x	—
BEQ, LBEQ	Branch if equal	—	—	—	—	—	—	—	x	—
BGE, LBGE	Branch if greater than or equal (signed)	—	—	—	—	—	—	—	x	—
BGT, LBGT	Branch if greater (signed)	—	—	—	—	—	—	—	x	—
BHI, LBHI	Branch if higher (unsigned)	—	—	—	—	—	—	—	x	—
BHS, LBHS	Branch if higher or same (unsigned)	—	—	—	—	—	—	—	x	—
BLE, LBLE	Branch if less than or equal (signed)	—	—	—	—	—	—	—	x	—
BLO, LBLO	Branch if lower (unsigned)	—	—	—	—	—	—	—	x	—
BLS, LBLS	Branch if lower or same (unsigned)	—	—	—	—	—	—	—	x	—
BLT, LBLT	Branch if less than (signed)	—	—	—	—	—	—	—	x	—
BMI, LBMI	Branch if minus	—	—	—	—	—	—	—	x	—
BNE, LBNE	Branch if not equal	—	—	—	—	—	—	—	x	—
BPL, LBPL	Branch if plus	—	—	—	—	—	—	—	x	—
BRA, LBRA	Branch always	—	—	—	—	—	—	—	x	—
BRN, LBRN	Branch never (3, 5 Cycle NOP)	—	—	—	—	—	—	—	x	—
BSR, LBSR	Branch to subroutine	—	—	—	—	—	—	—	x	—
BVC, LBVC	Branch if overflow clear	—	—	—	—	—	—	—	x	—
BVS, LBVS	Branch if overflow set	—	—	—	—	—	—	—	x	—

MISCELLANEOUS INSTRUCTIONS — Addressing Modes

Mnemonic(s)	Operation	Implied	Immediate	Direct	Extended	Extended Indirect	Indexed	Indexed Indirect	Relative	Relative Indirect
ANDCC	AND condition code register	—	x	—	—	—	—	—	—	—
CWAI	AND condition code register, then wait for interrupt	—	x	—	—	—	—	—	—	—
NOP	No operation	x	—	—	—	—	—	—	—	—
ORCC	OR condition code register	—	x	—	—	—	—	—	—	—
JMP	Jump	—	—	x	x	x	x	x	x	x
JSR	Jump to subroutine	—	—	x	x	x	x	x	x	x
RTI	Return from interrupt	x	—	—	—	—	—	—	—	—
RTS	Return from subroutine	x	—	—	—	—	—	—	—	—
SWI,SWI2,SWI3	Software interrupt (absolute indirect)	x	—	—	—	—	—	—	—	—
SYNC	Synchronize with interrupt line	x	—	—	—	—	—	—	—	—

sembly-language programs, as you can assemble the final program as a series of small modules that have no trouble fitting into memory with the editor/assembler/debug package.

Advantages and Disadvantages Of Assembly Language

Assembly language is extremely fast, typically dozens or hundreds of times faster than the equivalent Basic or other higher-level language. It is often more compact than a high-level language, using probably only half as much memory in typical cases.

Seems too good to be true! Are there any disadvantages? Yes, and they're considerable.

First, Assembly language is hard to use. Even with in-memory assemble and built-in debug packages, it takes four or five or more times longer to write the same code as in Basic.

In addition, you've got to learn the Assembly language, which probably takes about 1,000 hours for your first Assembly language and 100 hours for each successive Assembly language. To a certain extent, once you've learned one Assembly language you've learned them all, because you're also learning specialized techniques and algorithms with your first Assembly language. These techniques are built into Basic —things like conversions between ASCII and binary, printer drivers, and table searches.

Should you do Assembly-language programming? If you can afford the time to learn that first language, if you like to hack away at a bit level, and if you like glory, by all means learn Assembly language! Let me explain that last part: Most significant applications programs are written in Assembly language. You won't find many VisiCalcs or Electric Pencils in Basic!

Even if you don't plan on writing a 4K Assembly communications package, though, Assembly language might be worth your while. As you saw from the first part of the article, it's not too difficult to write simple Assembly-language programs and interface them to Basic to provide short, efficient code. Give it a try. ■

```
                      00100 ****BUBBLE SORT DEMO - COCO***
                      00110 *SORTS ONE-CHARACTER ENTRIES IN FIRST 256
                      00120 *BYTES OF SCREEN
0000 108E 0000        00130 BUBSRT  LDY    #0      Y IS "CHANGE FLAG"
0004 8E   0400        00140         LDX    #$400   START OF SCREEN
0007 A6   84          00150 BUB010  LDA    ,X        GET I ENTRY
0009 E6   01          00160         LDB    1,X       GET I+1 ENTRY
000B A1   01          00170         CMPA   1,X       COMPARE I TO I+1
000D 23   08          00180         BLS    BUB020    GO IF EQUAL OR LESS
000F 108E 0001        00190         LDY    #1        SET SWAP FLAG
0013 E7   84          00200         STB    ,X        SWAP ENTRIES
0015 A7   01          00210         STA    1,X
0017 30   01          00220 BUB020  LEAX   1,X       SCREEN ADDRESS+1
0019 8C   04FF        00230         CMPX   #$4FF     DO FOR 254 PAIRS
001C 26   E9          00240         BNE    BUB010    GO IF NOT DONE
001E 108C 0000        00250         CMPY   #0      TEST Y=0 OR 1 FOR CHANGE
0022 26   DC          00260         BNE    BUBSRT  SWAP, ANOTHER PASS
0024 39               00270         RTS            RETURN TO BASIC
          0000        00280         END
00000 TOTAL ERRORS

BUB010   0007
BUB020   0017
BUBSRT   0000
```

Program Listing 8. Bubble-sort Demo for the Color Computer

William Barden, Jr. is a computer consultant with nearly 20 years programming experience. He can be reached at 28122 Orsola, Mission Viejo, CA 926920.

Bibliography

Z80 Assembly Language (Models I, II, III)

Z80 Microcomputer Handbook, William Barden, Jr., Howard W. Sams, 4300 W. 62nd St., Indianapolis, IN 46268. General introduction to Z80 hardware and software.

TRS-80 Assembly-Language Programming, William Barden, Jr., Radio Shack. General introduction to Assembly language on the TRS-80 Models I and III.

More TRS-80 Assembly-Language Programming, William Barden, Jr., Radio Shack. Beginning and intermediate information on Model I and III Assembly-language programming. Contains a Morse-code program and a tic-tac-toe artificial-intelligence program.

TRS-80 Assembly-Language Subroutines, William Barden, Jr., Prentice-Hall Inc., Englewood Cliffs, NJ 07632. Contains 65 useful subroutines that are relocatable anywhere in memory and may be easily interfaced to Basic; it also includes tutorial chapters.

Practical Microcomputer Programming, The Z80, Walter J. Well, Northern Technology Books. Extremely well written and entertaining book! Be warned, however, Walt uses non-standard Z80 mnemonics.

Zilog Z80 Assembly-Language Programming Manual, Zilog. *The* definitive book on Z80 Assembly language.

Z80 Assembly-Language Programming, Lance Leventhal, Osborne/McGraw Hill, 630 Bancroft Way, Berkeley, CA 94710. Somewhat hardware oriented towards general purpose Z80 interfaces (not Radio Shack computers).

TRS-80 Assembly Language Made Simple, Earles L. McCaul, Howard W. Sams. Excellent TRS-80 Model I and cassette-oriented book with ROM calls.

8080/Z80 Assembly Language, Alan R. Miller, John Wiley and Sons, 605 3rd Ave., New York, NY 10016. Many listings and examples. Not Radio Shack oriented.

TRS-80 Assembly Language, Hubert S. Howe, Jr., Prentice-Hall. Includes disk information and calls.

6809E Assembly Language (Color Computer)

Color Computer Assembly-Language Programming, William Barden, Jr., Radio Shack. Introduction to CoCo Assembly-language programming.

MC6809 Preliminary Programming Manual, Motorola. *The* definitive book on 6809E Assembly language.

6809 Assembly-Language Programming, Lance Leventhal, Osborne/McGraw Hill. Somewhat hardware oriented for general 6809E interfaces (not necessarily CoCo).

✓See List of Advertisers on Page 563

Word Processing Guide

by Dan Robinson

Word processors ease the task of producing letters and manuscripts, but you must find one that fits your system and your needs.

The first time I heard the term "word processing," an image of words linked together like little sausages leaped to mind. In those days one could acquire a Wang for a mere $15,000. Besides the high price tag and the standard office jokes (How's your Wang today?), early word-processing machines were about as difficult to control as Three Mile Island. They were often sold with a three-month instruction course for the operator, their features were limited, and they could do nothing more than make words into sausages.

Then about a half dozen years ago, Michael Shrayer came out with his pioneer Electric Pencil, which gave the first word-processing capability to microcomputers. Since then, programmers have engaged in an unending contest to excel one another in adding more exotic features to word-processing programs.

How many times have you wished you could change a single word to make a business proposal more effective but didn't want to take the time to retype the document? Or how many times have you found yourself typing the same letter over and over again?

A word processor differs from a typewriter in its editing and printing. Most corrections are made by moving the cursor to the desired location and typing over the text. The cursor is usually moved by the arrow keys, although some word processors have commands that move the cursor forward or backward by word, sentence, paragraph, or to a specific character or symbol.

Since text is seldom short enough to fit on the 16 lines of the TRS-80's monitor screen, the word processor includes commands to scroll the text forward and backward into the viewing area. Some permit a quick jump to the beginning or end of the text, and some move one screen at a time.

Entire blocks of text can be deleted, duplicated, or moved to another location. Some programs accommodate several blocks at one time, allowing exchanges or a total reordering of the text.

The global-search function locates a specified string of characters. If a standard business letter has a string of NAME, the search function will find it each time it occurs. An expansion of this is the search-and-replace function that locates all occurrences of a string and replaces them with another string. Thus, NAME can be changed to Mr. Jones throughout the text.

If a word is too long to fit on the current line, wraparound places it on the next line of the video screen. However, some programs support conditional, or soft, hyphens, so if the entire word doesn't fit on a printed line, the word is broken at the conditional hyphen location and a hyphen is printed. If the entire word fits on the line, the soft hyphen is ignored. The width of the video display can be changed so the text is seen as it appears in print. This is useful when data is presented in columnar form or when hard hyphens are used. Tabs can be set to align columnar data.

Unbreakable spaces, such as between a person's initials and his name, can be supported. If an unbreakable space is specified, the entire name appears on the next printed line if it cannot fit on the current line.

A header appears at the top of each printed page, and a footer is at the bottom. They are useful as titles and can contain such information as the report date or the department issuing the document. They often include chapter headings and page numbers. Usually, an option exists to print them on odd, even, or all pages, much as they might appear in a book. Formats can be changed in conjunction with headers or footers to adjust the margins.

Standard paragraphs can be inserted from other files to make repetitious typing unnecessary. Some programs can chain files so a large body of text, such as a technical manual, can be printed without interruption of page numbers or without continuously specifying headers, footers, or print formats.

If supported, a specified number of

blank lines appear between paragraphs, and the new text is indented. The program can include reverse indentation that inwardly aligns a title or number with the adjacent block of text, as required in an outline. Since ending a page with the first line of a paragraph or beginning one with the last line is undesirable, some programs let you specify how many such "widow" lines are acceptable.

You can include comment lines in the text as notes or instructions to the operator, but they are not normally printed. Some programs support limited editing or input from the keyboard at print time to provide a fill-in-the-blank capability.

The program controls the appearance of the printed text, so you can specify items such as the number of lines to be printed on each page, the width of text, and single or multiple spacing. You can set left and right margins, as well as the number of blank lines at the top and bottom of each page. Most programs offer right justification, where additional spaces are added within the line to provide even margins on both sides. Some also allow an even margin on the right with a ragged left edge, as in some poetry. Or, the text may remain unjustified to give it a personalized look. A few programs take on the ambitious task of justifying proportional font printing, and some center a line or block of text horizontally as well as vertically.

Word-processing programs always support a parallel (Centronics) printer. Some also function through the RS-232 port to operate serial printers or work with the popular Small Systems Software TRS-232 printer driver through the cassette port. Some programs support control codes that use the special abilities of certain printers, such as condensed printing, italic or double-wide fonts, bold face type, or graphic characters.

Some word processors, such as those written in Basic, are limited to a specific number of characters on each line, and you must edit one line at a time. Usually, these are tape-based programs aimed at 16K computers, and they support few features. They are good for occasional or light use, such as preparing a club newsletter.

Most programs are character-oriented with a constant stream of letters and symbols. Since a main purpose of word processing is conserving time in written communication, most programs are written for disk systems. Some of these are displayed on the screen in a formatted presentation exactly as they will appear in print.

Radio Shack's Scripsit was one of Tandy's earliest and best efforts. Produced in both cassette and disk versions, its shortcomings have been overcome by several patch programs. One of the best is Flextext, which sends codes to smart printers to change type fonts, produces subscripts and superscripts, provides boldface headings, and so on. Qwerty, another Scripsit patch, creates an index and table of contents for lengthy documents.

Electric Pencil is probably the easiest word-processing program to use. It is inexpensive, because it offers few features in its stripped-down model; additional features can be purchased separately.

CopyArt offers all the standard features plus unique graphics capabilities that other programs can't equal.

Lazy Writer and Newscript offer the largest range of features, as well as the highest price tags. Lazy Writer, for example, lets you delete by character,

word, sentence, or block, or to a specified character, as well as from the cursor to the top or bottom of the file. Newscript automatically creates an index. Both of them let you program a number of keys with a series of commands you select to invoke their many features.

All programs mentioned are well documented and provide good after-purchase support.

When selecting a word processor, consider the availability of support software. Some word processors are compatible with programs like Special Delivery, which personalizes a form letter and integrates it with a mailing list. Special Delivery sends out a "personal" letter to everyone on your list.

Other programs like Electric Webster or Hexspell check for spelling errors and automatically correct the document.

Grammatik makes writing more effective by identifying poor style in a text file. It alerts you to the repetitious use of a word, as well as capitalization and punctuation errors. You can structure Grammatik to keep your computer jargon from creeping into communications with those who don't understand it.

As far as hardware is concerned, you need either a daisy-wheel or dot-matrix printer. Although the price of daisy-wheel printers is coming down, they still cost half again as much as dot-matrix printers, and they operate at about one-fifth the speed. The strong point of a daisy wheel is that the final text doesn't look like it comes from a computer.

With the newer model dot-matrix printers, the reader has to look closely to see that the text is a computer product, although the added quality results in reduced speed. Be certain that the printer has descenders so letters such as y and g won't appear to be pushed upwards. If you need graphics get a dot-matrix printer.

You'll need a disk drive for storage if the use of your word processor is heavy or complicated, or if labor is costly. Two drives are an advantage, since the advanced word processors load in modules to control the entry, edit, storage, and printing functions, and these modules must be present in a drive.

To save time, add a print buffer to your system. Then, your computer is available for use while the printer is busy with the text. The computer sends the formatted text at high speed to the buffer where it is held until the slower-speed printer is able to accept it.

With a properly equipped TRS-80 and a word processor like Electric Pencil 2, you can compose and edit the text; check it for proper spelling; identify poor writing style; print the document with data inserted from the keyboard, from files, and from a data base; and use the computer for other purposes while the document is being printed.

Exotic capabilities are not for everyone. Added features mean added cost, less text memory, more complex commands, and slower operation. The word processor that can process a multi-chapter technical manual or a graduate school thesis is not necessarily useful for someone who has a large volume of one or two-page letters to produce. Tailor your choice to meet your special requirements. ■

Dan Robinson (1625 Higgins Way, Pacifica, CA 94044) is the author of 80 Micro's *September 1982 special section on word processing.*

The Data Base Explained

by Wynne Keller

Introduction to data bases—everything you need to know explained in plain, simple English

Most complex computer programs include a data-base manager. Data-base managers (DBMs) maintain a file or group of items that are related. The DBM performs clerical office functions such as add, change, delete, sort, search or print for any item or group of items in the file. It is fast and efficient, but it cannot do everything. Generally, a DBM manages a file but cannot perform massive changes. For that, you need an auxiliary program.

Data-base managers are all-purpose programs and are useful in business and home environments. To choose the right data-base manager, you need to understand DBM terms and evaluate the program's intended use.

Types of Data-base Managers

There are two types of DBMs: in memory and random access. In-memory programs hold all the filed items in computer memory at one time. They can be cassette or disk based. Random-access programs are always disk-based; they bring each item into memory as needed, so random-access files can be larger than in-memory programs that are limited to smaller files.

Some programs try to incorporate the best of both types. In-memory programs are faster, because they don't need to access the disk each time you want an item. These programs allow you to divide the file into small sections that are retained in memory and written to disk as groups, and can be merged with other groups as the need arises.

In-memory programs are best for small files; even with 48K, a typical application holds only 200 items. However, large-capacity personal computers such as the IBM increase the usefulness of in-memory programs. A 128K IBM can hold three times as many items in memory as a 48K computer, so you can accomplish large projects with an in-memory DBM.

Because in-memory programs record the whole file as a unit, any system crash affects a large number of items. A random-access program is usually safer because each item records back on disk after access.

Selecting a Program

The three most important DBM terms are file, record, and field. Files are items in the data base. The record is one item in the group.

In a data base organized for household inventory, one item might be a sofa. You can record several things about the sofa: its price, purchase date, and present value.

Each piece of information about the sofa is a field. You enter the relevant field information for each record in the data base. If this data base were a card file, the entire box is the file, each card is a record, and on the card are three data fields about every item.

DBMs differ greatly in capability and price. Elementary versions can cost as little as $20, while sophisticated programs are $300 or more. The following descriptions of important features can help you choose the DBM best suited for your needs.

Capacity

Estimate the number of records in the file and the amount of information needed for each record. Every letter or number counts as one byte. In the household inventory, the name-of-item field (chair, sofa, and so on) will need perhaps 10 bytes. Round the price field to the nearest dollar, and if no item is more than $999, three bytes would be sufficient. The date of purchase could be the last two digits of the year, using two bytes, or five bytes to write it with the month and a slash, 12/82. Adding these together shows that each record needs 18 bytes. If there are 100 items in the house, only 1,800 bytes are required, easily within the capacity of a cassette-based, in-memory DBM.

On the other hand, a business computerizing its mailing list might find they will exceed the disk's capacity. A typical mailing list requires 115 bytes per record. On a two-drive system, the program resides on one disk and the data resides on another. But, even using the whole disk for data, more than 700

Become a "Crazy Painter" and create a masterpiece ...if you can.

A mischievous puppy, snakes and poisonous turpentine buckets force changes in your painting. In higher skill levels, "paint eaters" start chewing up your work. Suddenly, everyone's a critic. Can you overcome them?

And on top of it all, in the *Exclusive Challenge Mode,* you have to catch an army of runaway puppies. Crazy Painter is *Joystick compatible.*

$15.95 tape, $19.95 disc plus $1.50 shipping and handling charges. Indiana residents add 4% sales tax. Master Card or Visa orders accepted. Also available at your software dealer.

The Cornsoft Group ✓319

6008 N. Keystone Ave., Indianapolis, IN 46220 • (317) 257-3227

The Official
FROGGER™
by SEGA®
The popular arcade game for TRS 80® Model I & III
TRS 80® Model I & III
$19.95 Tape
$22.95 Disk
Joystick compatible. Frogger is available from your local software dealer or order direct.
All orders by VISA, Mastercard or check add $1.50 shipping. Indiana residents please include 4% sales tax.
The Cornsoft Group
N. Keystone Ave., Indianapolis, IN 46220 (317) 257-3227
designate trademarks of Sega Enterprises, Inc.
© 1981 Sega Enterprises, Inc.
TRS 80® is a registered trademark of the Tandy Corporation.

Includes exclusive challenge mode.

Bounceoids come crashing down from space — and rebound across the screen. Blast them or they will smash you. You have a shield but its lifespan is limited so use it sparingly.

The commotion will attract alien natives with poison darts, off-world snakes and shaking bugs. Be on your guard. The longer you survive the harder it is to simply exist. Bounceoids start dropping from the sky in tiny clusters or with constantly changing outer dimensions.

Reach the exclusive challenge level and face the flying space flock in a tense challenge of strategy, coordination and targeting skills. Joystick compatible.

$15.95 tape, $19.95 disc plus $1.50 shipping and handling charges. MasterCard & Visa orders accepted. Available at your software dealer.

The Cornsoft Group ~308

6008 N. Keystone Ave., Indianapolis, IN 46220 • (317) 257-3227

records can exceed the capacity of a Model I disk. If the program is not capable of spanning drives, it is worthless for this application.

Speed

A slow DBM is a frustrating experience and can even be unusable. Many of the program's aspects affect the speed: the method of data entry and change, the method of saving data, and the type of search and sort.

Data entry can be tedious in some programs. Ideally, a fast typist should be unable to overtype. Frequently, however, the typist must slow down and check the screen to be certain the program recognized keys that were pressed.

An important feature for increasing data-entry speed is a field-repeat key. This allows the typist to press one (or sometimes two) special keys to repeat any field's contents from the immediately previous record. For example, if you're typing addresses and several in a row list New York as the state, New York can be re-entered with one keystroke. The key that performs this function varies from one program to another, and many data bases don't have the feature at all. It is worthwhile to look for the program that does.

The ability to change entries rapidly is also important. If you make an error when typing the record, you should be able to correct the error before moving on to the next record. A nondestructive cursor is the easiest method for changing data. Nondestructive means the cursor blinks over, but doesn't destroy the letters beneath it. The cursor can be moved to any location on screen with the arrow keys and changes are made by typing over the error, or the text can be opened for insertions.

Another method of changing entries is numbered fields. To make a change, you type the number corresponding to the field in which the error occurs. The cursor then jumps to that field and the change can be made. The slowest method places the cursor at the end of the first field whenever you specify change mode. To change any field, press the enter key repeatedly while the cursor jumps down one field at a time until it is on the field where the error occurs. This is a tedious process, especially if the field to be altered is near the bottom of the screen or many records are to be changed.

The method of saving or reading data to or from the storage medium, called I/O (input/output), also affects the speed of operation. Cassette users planning DBM applications must first consider whether the project is possible on cassette. The length of time needed to save data on cassette makes a DBM for files larger than about 100 records impractical. Model III users are often unaware that the speed of cassette dumps for data, as opposed to programs, is only 500 baud. In addition, each 256-byte segment of data has a leader on the tape, which also slows data saves. You can prevent this leader with machine code, and some cassette DBMs advertise this feature.

If the project is not worth the price of a disk drive but is too large for cassette, consider the Exatron Stringy Floppy. The ESF saves data at 7,600 baud, which means 150 records are saved in about one minute.

Disk users also should consider the speed of data I/O. With in-memory DBMs, the data loads at the beginning of a session and saves at the end. This is convenient and is the main reason for the popularity of in-memory systems. The random-access DBM loads one record at a time and then saves the record as soon as you are finished with it. Since the disk turns on for each save, there are a few seconds of idle time for the operator with each I/O. When adding records, you should not record the data until you have corrected any errors. I know of one program that does not allow error correction during input; it sends the data to the disk as soon as you finish the last field, causing double disk I/O time.

Disk I/O for random-access programs sounds more tedious than it is in practice. When only a few records will be accessed, random access is faster than an in-memory system because you can load a few records faster than the whole file. If you start out with an inmemory system, you can stretch it to handle larger and larger files. Perhaps the entire mailing list was 200 records initially, but expanded to 400, so the file was broken into halves, then halves again. Switch to random access before the file becomes too large; it's easier than shuffling and merging files that have outgrown the program.

Searching/Sorting

The type of search and sort is vital to program speed. A fast search is more important because sorting can be performed during your absence. The slowest search is sequential; the program examines each record to see if it matches, in order through the file. A binary search divides the file in halves with each pass, and finds the needed record faster. One of the fastest is an index; an index is a separate short record number file that keeps track of the major file. A record in the index contains the location or record number of its matching record in the major file.

Because record length in the index is short, the index can be read quickly (in some programs it is kept in memory). Normally, the operator chooses the field used in the index. Some programs allow the indexing of several fields. The index increases the speed of search and sort functions. Its only disadvantage is apparent if there is a machine or power failure.

Some programs rebuild the index on command. This is a good feature, because you can repair the index if it is damaged. Other programs update the index automatically after changes have been made or after 100 records have been added to the file. This type is vulnerable; if an equipment failure occurs after adding 50 records, all 50 would be lost. Even though the records are safely on the disk, they have not yet been added to the index, and without the index they cannot be found.

An index increases sort speed. The fields to be sorted are read from the disk; when the sort is complete, the index is rebuilt to reflect the new order. The index is then written to disk, and the records accessed in sorted order, even though none of the major records have been moved from their former position on the disk.

If the sort is in machine code, the sort speed increases, but this is not the only factor in sort speed. Disk access can be the culprit when sorts continue for hours. A number of programs sort by reading two records, comparing them, then switching their order and rewriting them to disk. The disk runs continuously, and it is not uncommon for the sort to fail because the disk became worn before the sort was finished.

Reports

Designing a report with most DBMs is time consuming, because each user wants something different, and the programs allow considerable flexibility. Once designed, a report format will be used many times. Therefore, the best programs save the report design in a file on disk, and it can be recalled and used without further effort. If a program cannot save report formats, it will be limited in report capability.

Some features to look for are:

- Choice of record numbers/no numbers
- Multiple lines per record
- Mailing-label format
- Saves report format
- User-specified subtotals and/or totals
- User-specified titles
- Wraparound suppression
- User-specified paging

It is standard on most DBMs that the user can specify which fields to print and which records to print. The way this is done is by no means standard.

If you plan to use a word processor to write form letters from a mailing list, make sure the word processor and DBM you plan to use are compatible. Some DBMs interface with any word processor file; others require a specific program.

Calculations

Many programs offer one or more formula fields, which allows you to create a formula that uses values from other fields. You could set up an inventory to calculate discounts or markup with a formula field that takes the price field and multiplies it by some other value. Some DBMs perform these calculations when the record is typed. Other DBMs show nothing on the screen during data entry, but display the computed values of formula fields later when you recall or print the record. Formula fields are vital in many business applications.

Setting Up a File

Using the above guidelines, you select a DBM. How do you put it to use? After reading the manual, you may understand how to use the program, but not how to set up the file. Most manuals assume some familiarity with the organization of DBM files.

The first step is to write down the information catagories being kept with your current manual system. If there is no current system, write down all the types of information you would like to keep. I will use inventory management as an example, as this is a fairly complex, but common application of a DBM.

The following are possible categories:

● Name of Item	20 bytes
● Supplier	25
● Supplier Address	20
● Supplier City, State, Zip	28
● Supplier Code	10
● Store Code	10
● Salesman	20
● Supplier Phone	14
● Price Item	6
● Cost Item	6
● Quantity on Hand	3
● Quantity on Order	3
● Reorder Point	3
● Reorder Date	8

Next, consider typical information you would place in these categories. Calculate the maximum number of characters that would be stored in each category. In the case of name categories, you can do this by estimating the average size of an entry and adding a few characters. Some name entries can be abbreviated later if they are too long to fit.

You must calculate the byte count for numerical and date entries with greater precision. The cost field, for example, must contain enough bytes for the greatest cost you will ever encounter. If you allow six bytes, the data entered can never exceed 999.99 (the decimal point counts as one byte). A date field is normally MM/DD/YY, or eight bytes. Typical byte counts are already indicated in the sample inventory above.

After you have a byte count for each field, add them up to determine the total number of bytes per record. The example inventory would use about 200 bytes. If the total per record is over 256 bytes, you may need to eliminate some characters, depending on the program being used; some programs don't allow a single record to exceed 256 characters.

Even if the bytes per record are within the program's limits, the number of records that will fit on the disk is seriously reduced with large records. In the sample inventory, the vendor's address is probably superfluous. The ad-

dress, phone number, and salesman's name could be kept in separate file consisting of all the addresses your company needs.

It's time to fine tune the DBM layout. The more sophisticated programs allow special types of fields. Alphanumeric fields can contain any character. Numeric fields may only have numbers and number-related characters. They may even have a fixed decimal point, if desired. Some fields can be "must fill" fields, which means the operator must enter something when typing the record. You can reverse date fields, so that on sorting they will be ordered YY/MM/DD. Fields may be protected, which means the data in them cannot be changed.

In data bases that use this searching method, choose a key field. The key field should be a field unique for every record. It would be useless to make the key field the state in an address file, because so many records will occur with the same state. Choose some type of ID-number field for the key field. In an inventory, this would be the part number.

Next, design the screen. Sophisticated programs give the user complete control over placement of fields. Even programs that do not allow this can be helped with planning at this stage, because the order in which fields are typed will be the order in which they are presented on screen. Ideally, all "must fill" fields should be first on screen, so the operator does not have to step through empty fields to reach the ones for which there is data.

In some programs, the fields aren't numbered and a change in a field near the bottom of the screen means you have to keep pressing enter to position the cursor at the proper field. This is not a problem if the cursor movement is rapid. If it is slow, positioning the cursor can be time consuming. If you use such a program, position any fields that will be changed often near the top of the screen. A few programs require fields that will be sorted consecutively to be adjacent to each other, with the major sort field first. If using either of these programs, make sure you type the fields in the correct sorting order.

Once you initialize the DBM and design the input screen, it is time to begin using the program on a trial basis. Enter perhaps 25–50 records; then make changes, sort them, and use the search function. If data entry or changes are slow, would a change in screen design help? Is the key field inappropriate? Design a few reports and print the records. Will all the fields you want on one line fit?

It is typical during this stage to find that some vital field was overlooked, or a field is too short, or too long. For example, using the inventory described earlier, trial runs might show a need for a formula field to calculate price as a percentage of cost. Remember, most DBMs do not allow any changes to a file after initialization without losing all the records. A few programs allow changes, such as adding a field, but this is difficult and time consuming. Be sure to initialize the file the way you want it, because with most programs everything must be typed in again if changes are made.

If you find the program inadequate during this stage, it is better to buy another now than to type a complete file into an unsuitable program. Changing to a better program later requires retyping the file or hiring a programmer to translate it.

Manipulating the File

Putting the file to use quickly reveals why DBMs are so popular. The sophisticated search and select capabilities are perhaps the most exciting to use.

To use selection features, you'll need an understanding of Boolean relationships. Most programs understand the following relationships: equal (=), greater than (>), less than (<), not equal, greater than or equal, and less than or equal. These concepts are familiar to most people. The harder part is the connective for the search (and, or).

Suppose two search criteria were desired: price less than a dollar, and supplier name XYZ. You would choose the price field as the first search field, and specify less than (<) as the relationship and 1 as the amount. Then specify the connection to the next field. In this case, you want every item under a dollar *and* supplied by XYZ. If you said *or*, you would get every item under a dollar from every company along with every item from XYZ regardless of price. Then enter the next search field (supplier name), along with the relationship, equal, and the name being sought, XYZ.

Suppose you receive a shipment from XYZ containing five different items. You want to update the record for each item by changing the quantity-on-hand field to reflect the amount received plus the amount formerly on hand. In addition, the amount on order needs to be reset to zero. To do this, you could search for each of the five items by name or product number, and change each record. This works well for five or ten items, but if hundreds of items were received, it is too time consuming.

The limits of the all-purpose DBM have just been reached. What you need now is an auxiliary program. Such a program would, in this example, accept data such as part number and quantity received for as many items as you wish to enter. Then, the program automatically accesses the DBM file and updates every record affected by the shipment. It sounds simple, but you need a programmer to write the auxiliary program.

DBMs are a compromise. You can use a DBM program for multiple tasks, thus saving the money it would cost to buy several different specialized programs. But a DBM inventory cannot hope to have the sophistication of a specially written inventory, unless program modules are added. Adding such modules does not have to be expensive but is a factor to consider. Many applications will not need extra programming, and if the DBM can computerize more than one file for you, it will almost certainly pay for itself. ■

Wynne Keller can be reached at Downeast Digital, RD#1 Box 4130, Solon, ME 04979.

Insure Your Computer

by Thomas McDowell

Remember to purchase insurance for your computer; you'll need it when lightning strikes!

If lightning hits your house, it can be a frightening and unsettling experience. It can also damage your computer. Last year, lightning struck my house, ruining several appliances, including my TRS-80 Model I.

When I filed the insurance claim, I learned some useful information about computers and insurance. My experience should help you if you find yourself in a similar situation.

Is It Covered?

If your computer is damaged, read your insurance policy to find out if it is covered. Your policy may not mention the computer, but it will mention lightning damage. Usually, if you have a good home-owner's policy, your computer is covered for this type of loss. If you use your computer for business, you will need a special rider for it, since a business computer is not covered under a home-owner's policy.

OK, so you think you're covered. What's next? Contact your insurance agent. Eventually, an adjuster will call you. He may want to inspect the damage, but since lightning damage is usually internal, he will probably tell you to have the computer repaired, if it is "economically repairable."

If it cannot be repaired, have the repairman make a statement to that effect. If the item can be repaired, get an estimate and report back to the adjuster. If you want the insurance company to pay for the repairs, the bill must clearly state that the item was damaged by lightning.

Economically Repairable Damage

Any item is repairable if the owner is willing to pay enough money. Insurance companies recognize this, and to limit their losses, they will limit the amount they will pay to repair an item to its present value. To compute depreciated value, my insurance company divides the age of the item by its accepted life, subtracts that value from one, and multiplies the difference by the current replacement cost of like items.

Some insurance companies may use original cost instead of replacement cost. For most items in inflationary times, the replacement cost policy is best because it reflects inflation. Unfortunately, my TRS-80 cost nearly $200 more when I bought it than it did when lightning struck it.

The owner must establish the computer's age. If you do not have the purchase receipts, tell the insurance adjuster where and when you bought it, and he can check with the retail outlet. If you are putting in a claim for stolen equipment, especially equipment worth thousands of dollars, and you don't have receipts, you could be in trouble.

Be honest with your adjuster. He deals with people every day, and he's no fool. My adjuster believed me when I told him that I had one of the earliest serial numbered TRS-80s and had purchased it in January 1978. My credibility had been established, so he again believed me when I said that my 32K Expansion Interface was new.

Determining the expected life of my computer proved more of a problem. My adjuster asked me to get the expected life from Radio Shack. The technician at the local computer center, however, refused to give me an estimate for my computer. He told me it should last forever. Even when I told him that the insurance company would consider that unrealistic, he refused to make an estimate. (He stated that he could be liable to Radio Shack if he made policy for them, and stating an average lifetime could be construed as making policy.)

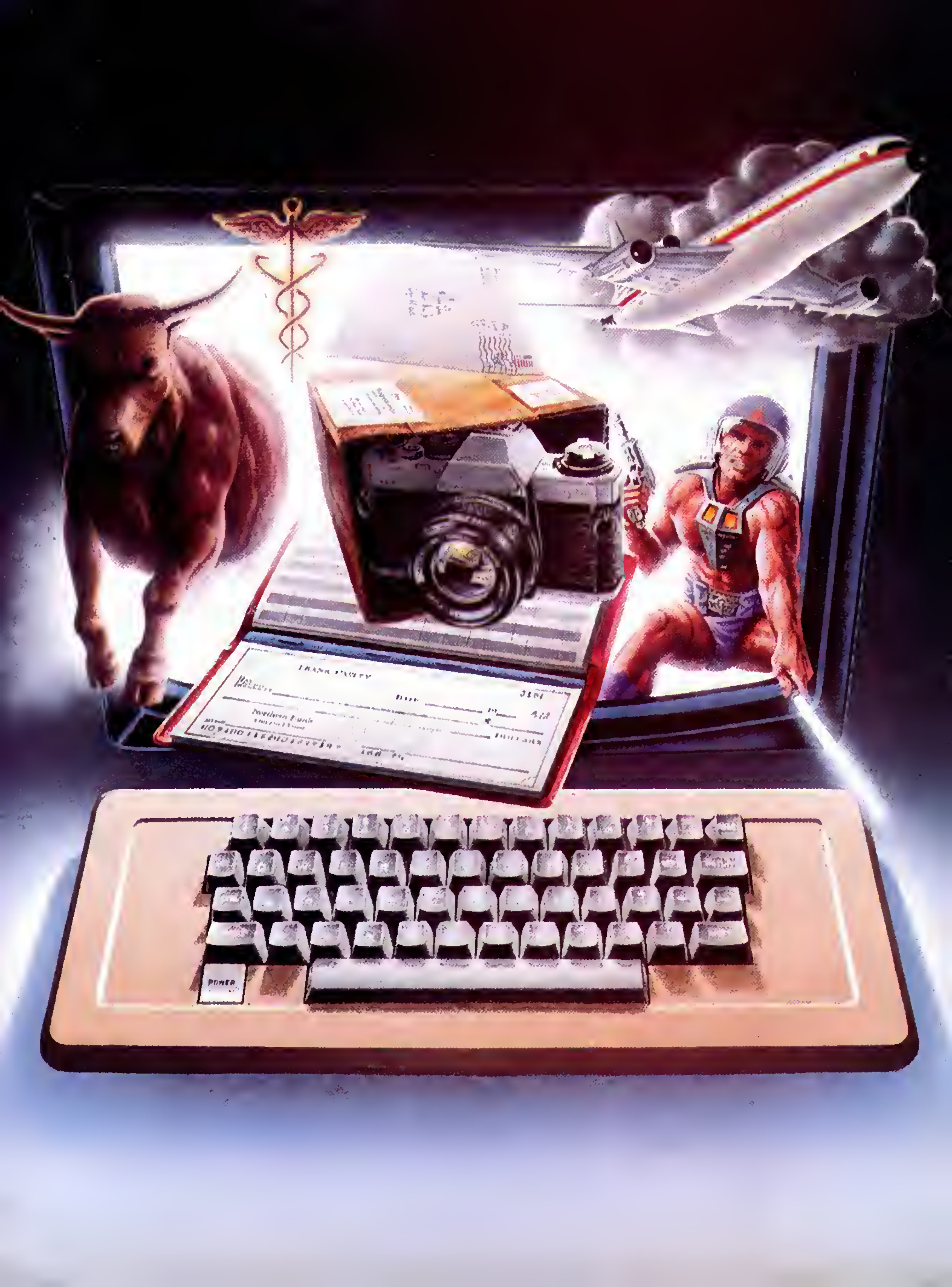
After some dickering, the adjuster and I agreed that my computer's life expectancy was at least that of a color television (10 years). He gave me a check for the depreciated value of my equipment, less the deductable, and I gave him the remains of the computer.

The adjuster told me that had my unit been stolen, I would have needed to prove ownership. The best way to do this is to have a receipt. Club registrations and owner manuals can also serve as proof.

Lessons Learned

The lightning that struck my house cost me over $400 for equipment replacement; without insurance, it would have been more than $2,000. To protect yourself from lightning or theft, remember three things: Unplug your equipment when an electrical storm is forecast; make sure you have insurance for your equipment; and save your sales receipts. You should also think about protecting your equipment from fire or other disasters. And what about your software? Storing back-ups off site is probably the best way to protect it. I hope you won't experience the disaster that I did, but you should take precautions, just in case! ■

Thomas McDowell is a data processor. His address is HHC 1st Support Battalion, APO New York 09137.


POWER

Disk Mysteries Revealed

by Michael F. Morra

Debating on whether to upgrade to a disk system or to keep your faithful cassette player? If you have any doubts, read this article.

Are you frustrated with the trials and tribulations of the cassette, but apprehensive about taking the big step to floppy disk? Or perhaps you are a seasoned floppy-disk user who is tired of paging through a half dozen magazines, texts or operating manuals before you find a satisfactory explanation of a particular subject.

This article attempts to correlate the existing information and reintroduce it in the form of a simple, slow-paced and (hopefully) comprehensive primer on the floppy disk.

At the start, I will be on the level of the non-disk TRS-80 user who is fairly conversant with the cassette-based system. I will assume some knowledge of binary/hexadecimal number systems, cassette data storage and the basic memory structure in the TRS-80. After laying the initial groundwork, I will move into a more advanced and detailed treatise concerning the Model I floppy-disk system.

With a Little Help from My Friends

This article owes its existence to the presence of various publications and the writers. (See the Bibliography.)

These publications will undoubtedly help those readers who demand an extremely detailed treatment of a specific question. There are many other sources that were not (or could not) be consulted during the writing of this article.

Most of these publications or products are available from the many advertisers in *80 Micro*. The following addresses may also be helpful:

Shugart Associates
475 Oakmead Parkway
Sunnyvale, CA 94086
(408) 733-0100

Western Digital Corp.
2445 McCabe Way
Irvine, CA 92714
(714) 557-3550
(For information on the FD 1771-01 chip)

Some of the Western Digital documentation has been republished in various places, including *Pathways Through the ROM.*

What Is a Floppy Disk?

A floppy disk is a form of *non-volatile* mass data storage for the TRS-80 and other microcomputers. Non-volatile storage (which also includes magnetic tape and read-only memory, or ROM) retains data without the application of external power. In contrast, random-access memory (RAM) is *volatile*—it requires power to continually refresh the stored data, and when the power is interrupted, the data is irretrievably lost.

Like cassette tape, a floppy disk is a scaled-down version of mass storage used in larger computers (minicomputers and mainframes). The granddaddy of cassette tape is the high-speed magnetic tape unit that employs open reels of wide recording tape. Large computers also utilize hard-disk storage, where data is stored on rigid metal disks coated with a magnetic material, much like the coating on recording tape. The computer industry modified disk storage to utilize flexible (floppy) disks, usually made from Mylar or similar material, with the same magnetic coating as before.

Briefly, here is a breakdown of the various components that make up a floppy-disk system:

- The *floppy disk* itself is the actual storage medium. (The disk includes both the coated disk and the protective jacket into which the disk has been sealed.)
- The *disk drive* physically manipulates the disk and, under system control, *reads* and *writes* data ("plays" and "records," respectively).
- The *disk controller* is the interfacing circuitry between the computer and the floppy-disk system.
- The *disk operating system* is the software that coordinates and executes all data *input/output* (*I/O*) to the floppy disk. It actually extends and enhances the ROM routines so that the floppy-disk system can operate properly. The software is usually called the *DOS*, which rhymes with "boss" or "dose" depending on whom you talk to.

Why Floppy Disk?

Let's take a look at the differences between cassette tape and floppy disk (vital statistics courtesy of Radio Shack's Computer Catalog RSC-6 and the *TRSDOS/Disk Basic Reference Manual*).

For both systems, the physical form of the stored data is the same: magnetized areas in a magnetizable coating. The data is both placed on the media and retrieved by a read/write (record/play) head in both systems. However, the method and the speed of data *access* is quite different.

Cassette tape is relatively exasperating when it comes to finding a given spot somewhere on the tape. The tape has to be manually jockeyed back and forth to locate the desired area. In contrast, floppy-disk systems can very quickly position the read/write head anywhere on the media to pick out a particular portion of written data. A floppy disk is even more versatile because of the data *format* on the disks, which allows us to locate and access very small units of data. This would be analogous to a stereo turntable that could automatically find and play a group of several notes anywhere on the LP disk within seconds.

Furthermore, floppy-disk systems have a much faster I/O rate than cassette systems, which makes the former even more powerful. Cassette I/O rates for TRS-80s are either 500 or 1,500 baud, which is equivalent to about 54 or 171 *bytes* per second, respectively. Floppy disk I/O rates are on the order of tens of thousands of bytes per second. In both cases, the figures are calculated for a continuous stream of data being read or written. An average access time would be the time elapsed for either system to find a given spot somewhere on the media. On a C-20 cassette (20 minutes running time), we could easily average one minute trying to find the spot and line the tape up fairly accurately. For a floppy disk, the time ranges from ¼–¾ second (and again, the disk access is fully automatic).

The disk usually contains more data than the cassette tape, too. Going back to our figures on cassette I/O, a 20-minute cassette can hold a theoretical total of 75,000–225,000 bytes of data, depending on the baud rate. Floppy disks can theoretically store 89,000–1,000,000 bytes, depending on disk size, data density, and whether one or two sides of the disk are used. (These figures do not include necessary system data, the so-called system overhead, which takes up some room on both cassette and disk.)

The floppy disk does have some drawbacks, however. Because of its precision and tight tolerances, it is more sensitive to the same problems that plague cassettes: motor speed variations, dirty or magnetized read/write heads, destroyed data from stray magnetic fields, physical contamination and mishandling of the media, power-line variations, and so on.

The major disadvantage to floppy disk is cost. To convert a system to floppy disk requires controller circuitry, at least one disk drive, miscellaneous cables and hardware, and the DOS software. Additional RAM may have to be added (the DOS occupies a good chunk of memory), and it may be desirable to have more than one drive hooked up. Also, floppy-disk-based software is usually more expensive than cassette software (but more powerful and versatile, too). The total system investment depends on the type and brand of hardware/software purchased, and where it was obtained (Radio Shack versus mail order, computer store, and so on).

Both systems have their place depending on the individual user. Cassette tape is an excellent entry-level mass storage device, combining ease of operation with low cost, and allowing the maximum amount of memory available for other tasks. Once a user becomes more intimately acquainted with the computer, though, the cassette can become the proverbial tail wagging the dog by hampering more advanced operations. At this point, it is a good idea to seriously consider the move to floppy disk.

The Nuts and Bolts of Floppy Disks

The disk (Fig. 1) is either 5¼ or 8 inches in diameter. The disk is free to rotate within a jacket, which is specially lubricated for that purpose. The exposed central ring of the disk is gripped by the motor assembly of the disk drive. The jacket also has openings (the *read/write notches*) on either side, where the read/write head contacts the disk surface. The *write-protect notch* is analogous to the break-off tabs on cassettes; if a disk's write-protect notch is covered with an adhesive tab, data can-

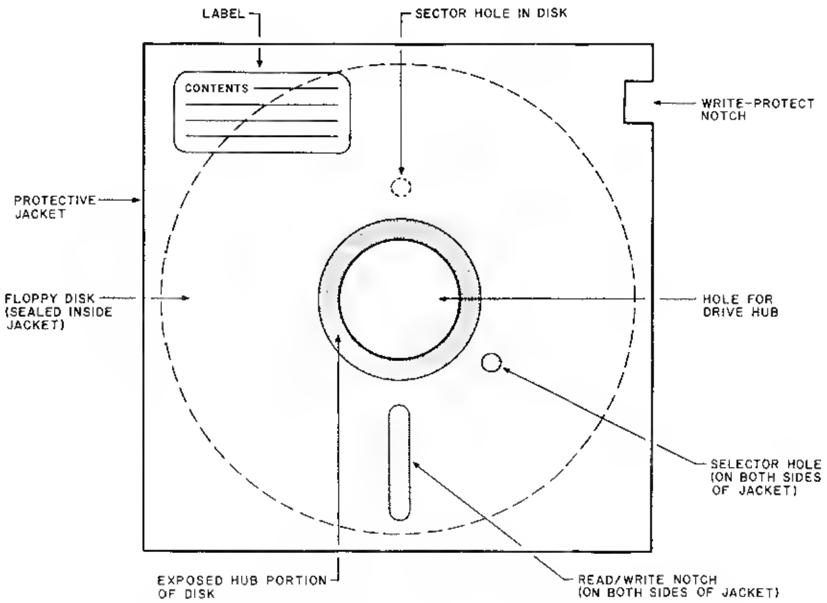


Fig. 1. A floppy disk. The actual disk is encased in a protective jacket

not be written to that disk.

The *sector holes* in the jacket and the disk communicates to the system the positioning of the disk, and enable the automatic location and access of the data. We will go into more detail about this process later.

There are various types of disks available. The following requirements are for those TRS-80 systems with all Radio Shack hardware and software. If your system includes any non-Radio Shack components, you may have to use a different type of disk.

The most obvious criterion is disk size. TRS-80s use 5¼-inch disks, except for the Model II which requires 8-inch disks.

Disks are certified as *single* or *double-sided*. Although both sides of the disk have a magnetic coating, sometimes only one side's coating is reliably applied and satisfactory for data storage. Most disks, whether single or double sided, have read/write notches on both sides, which means that it's possible to read/write data on a "bad" side (usually leading to various I/O errors). All TRS-80s use only one side of the disk. Double-sided disks work fine with no system modifications. However, you pay for a certified second side that the TRS-80 (unless modified) can't use.

Another specification is *single* versus *double density*. Certain systems have the capability of packing a larger amount of data in a given physical area on the disk. Because of this high data density, the magnetic coating must be more uniform and reliable to avoid loss of data. Except for the Model I, all TRS-80s require double-density media. (Although the Model I was set up for single-density disk I/O, double-density disks can be used with no system modifications. In fact, the double-density is probably worth the extra money for its higher reliability. Radio Shack, by the way, sells only double-density media for the Model I and other TRS-80s).

Disks are also offered in *soft* and *hard-sectored* varieties. The difference is the number of sector holes in the disk (*not* the jacket—see Fig. 1). Soft-sectored disks have one hole; hard-sectored disks have a continuous ring of holes around the large hub hole (but hidden under the jacket). All TRS-80s require soft-sectored media, and unless modified, will not function if hard-sectored media is used.

Be suspicious of media priced far below the average going rate. El Cheapo disks usually have relatively poor coatings that lead to all sorts of headaches with lost or inaccessible data.

Better disks usually have *reinforced hub holes*; you can see the double-thickness edge of the exposed disk at the hub opening. The increased thickness prevents wear or elongation of the hub hole, which may result in unreliable disk operation. Kits are also available to apply reinforcing rings to nonreinforced disks.

Disks should be handled in a very careful manner, owing to the sensitivity of the written data. The disks should be filed vertically (on edge) in their protective envelopes, away from heat and sunlight, and not compressed against each other excessively. Even very minute particles on the media can jam between the read/write head and the rotating disk surface, and some of the magnetic coating can be gouged out. These offenders include dust, cigarette ashes, and even the particles in cigarette smoke (another good reason to give up smoking). Disks should be kept away from stray magnetic fields, which can be produced by power cords, speakers, motors, tv and stereo receivers, video monitors, and so on.

When ready to shut off the computer, remove all disks from the drives. When writing on the disk label, use a felt-tipped marker; pencils, ball-point pens, and other hard-pointed instruments can mar the disk surface under the jacket.

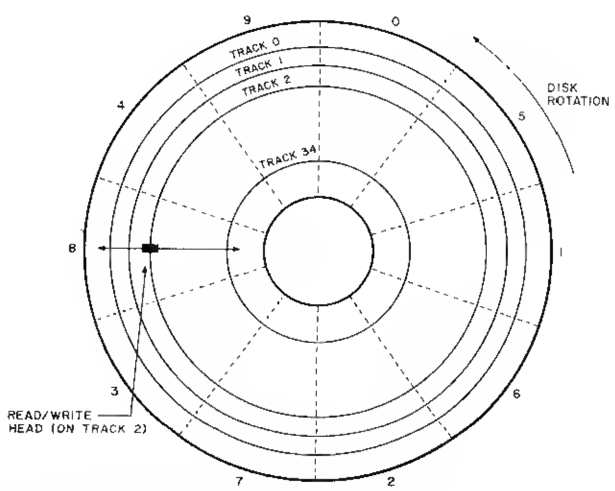


Fig. 2. A formatted disk. The number of tracks and sectors will vary, depending on the system. Note the staggered numbering of the sectors.

Data Here, Data There

Examining how data is arranged on the disk will allow us to define some important concepts and buzzwords for later use. For now, however, don't worry about how the data is accessed, or other specifics about the I/O process.

Figure 2 shows a working TRS-80 disk with the jacket removed. Data is laid out on the disk in ring-shaped, concentric *tracks*. As shown, the read/write head on the drive can move in or out along the radius of the disk, positioning itself over a desired track. As the disk rotates, the head "sees" all the data contained on the track.

In addition, the data on a track is organized into a number of *sectors*, as indicated. There is no physical gap between the sectors; certain data patterns that are part of the system information distinguish the various sectors.

There is an important concept here: Each sector contains both *system information* and *user data*. The latter represents the actual data that the user is normally concerned with. As you would guess, the system information is for the benefit of the system itself. It performs such housekeeping tasks as delineating and identifying sectors, verifying the data, and so on. As we will see later, the system bytes form a skeletal structure on the disk, into which the user introduces the data to be stored. In general, the system bytes are invisible to the user, unless a utility such as Trakcess is used to access them.

The inner tracks and sectors are physically much shorter than the outer ones, but they both contain the same amount of data. If you feed your TRS-80 bargain-basement disks, those dense inner tracks may give you headaches with lost data due to inconsistency of the magnetic coating.

Putting Data in Its Place

To get some idea of how the data and system information work together, we'll briefly examine disk *formatting*. Any blank disk must be formatted before data is placed on it.

DOS packages include a format utility to accomplish this.

When a disk is formatted, it is first tested for flaws by actually writing a data pattern onto it, and then reading it back and verifying it. If any discrepancies are detected, the system assumes that a defect in the media was the cause (the most likely possibility, but hardware bugs can also cause similar problems). The system then locks out the questionable tracks by writing information onto the disk that later instructs the system not to write data to those tracks.

Once this is accomplished, the system gets to the heart of the matter. First, it establishes track and sector boundaries by writing a pattern of system information to the disk. This information also identifies each individual sector so they can be accessed at will later on.

The next step, a very important one, is to establish a *directory* (usually one track long) on the disk. The directory contains status information such as the location of locked-out tracks, access limitations (passwords or other protective functions), and so on. Later, as data is written to the disk, the directory keeps track of where everything is, and allows the user (and the system) to find it again.

Depending on the system, other data may be written during formatting. In the Model I, for example, a *bootstrap* routine (or *boot*), which is necessary for system operation, is put on the disk.

Notwithstanding some different options that may be offered, the basic actions during formatting are the same for all systems: checking the media and then preparing it to accept data.

The Little Gray Box

The *floppy-disk drive* performs the same function as the cassette recorder in a tape system. It executes commands to access existing data or to write new data. However, all the disk drive I/O functions are automatically controlled by the system, as opposed to the manual button pushing required in cassette systems. This is yet another factor contributing to the power and versatility of the floppy-disk system.

The drive incorporates the following subassemblies:

- A *drive motor*, with a hub assembly, grips the disk at its center hole and rotates it at an accurately controlled speed. The hub locks onto the disk when the drive's *access door* is closed
- The *read/write head* is positioned over the read/write notch in the jacket. When data is to be accessed, the head is moved in and out along the radius of the disk by a *stepper motor*.
- The *write-protect switch* checks the write-protect notch on the disk.
- The *index/sector LED and detector* are positioned on either side of the disk at the sector holes. As the disk rotates and the sector hole lines up with the jacket holes, a flash of light falls on the detector, which then triggers logic circuitry to inform the system about the disk's position.
- *Logic circuitry* (either *control* or *read/write* circuits) coordinates all drive operations. Most circuits light an LED on the drive cabinet when the driver motor is running. *Do not* attempt to open the access door when the activity LED is lit!

Just as with disks, there are many variations of disk drives to choose from. Again, the following specifications are for unmodified Radio Shack TRS-80s.

Drives are supplied either as bare drives (not mounted in a cabinet, and usually without power supplies), or completely self-contained external units (with power supply and cabinet). There are also multiple-drive assemblies that incorporate two or more drives in one cabinet. The Model I and Color Computer each use up to four external drives. The Model III has space in its enclosure for two bare drives to be installed, and can support two more external drives. The Model II has one internal drive, and can support up to three additional drives. (Model IIs usually have a three-drive expansion bay for extra drives. The bay is a cabinet/power-supply unit with one drive installed, and two extra bare drives can be installed as needed.)

Some systems require that one drive incorporate *terminating resistors*, which provide a satisfactory electrical match in the disk interfacing. Usually, there is no difference in price between standard and *terminal* drives. For all TRS-80s (except the Model II), one of the external drives is usually a terminal drive.

On the Color Computer, the first drive purchased will be more expensive than additional drives. Similarly, on the Model III the first internal drive is more expensive than the second. In both cases, the *disk controller circuitry* is incorporated into the first drive. The Color Computer's first drive kit also contains a program cartridge containing the DOS and a cable for the drives.

The Model I is a different story; the controller is not found in either the computer or the first drive, but in the *expansion interface*, an additional piece of hardware needed before any drives are hooked up.

Your drives need *track capability* to match the system's track setup. The Model I is set up for 35-track I/O, the Model III and Color Computer are 40 track, and the Model II is 77 track.

As with disks, the drive must be able to handle the appropriate data density. Specifically, the read/write head should be certified to handle double density I/O (except for the Model I, which is single density). Most drives available today, except for the lowest-priced models, are capable of both single and double-density I/O.

You should use double-density-capable drives in the Model I with no system modification. In this case, you would not have to replace the drives if you later decide to convert the system to double density.

Tell Me What You Want

The disk drive by itself is a rather stupid piece of sophisticated machinery. Its only real capabilities are that it can merrily spin a disk at several hundred rpm and step its head in and out (as well as several reporting functions, like checking the write-protect notch or the sector hole). Obviously, some direction from hardware or software is needed to perform worthwhile disk I/O. The disk controller circuit is an important example.

The controller circuit usually consists of little more than a single IC and a small handful of support components. However, it tremendously expands the repertoire of floppy-disk functions and data transfers. For example, the Western Digital FD 1771-01 controller chip (used in the Model I) enables reading/writing of bytes, sectors or tracks, formatting and enhanced seeking/locating capabilities.

The controller is functionally invisible to the user since it acts under command of the DOS. Unless you try to design your own DOS or a program with its own disk I/O routines, you do not have to delve too deeply into the controller operation. For those of you who are interested, references include Western Digital's documentation on the controller chip and Bill Barden's text on disk interfacing (see bibliography).

DOS Is the Boss

At this point, we have covered the physical components that comprise a TRS-80 disk system. However, in this raw state it is capable of only rudimentary disk operations under direct opera-

tor command (since the ROM has no disk I/O routines as such). In order to make this conglomeration of expensive hardware sprout wings and fly, we need a piece of software known as the *disk operating system*, or *DOS* for short.

To quote from the Radio Shack TRSDOS manual:

> "An *operating system* is a master program that allows a complex computer system, including various input/output . . . devices, storage devices and programs, to interact efficiently and with apparent simplicity. The operating system makes sure that everything that has to be done gets done—and you don't even have to know what it is that 'has to get done.' "

As you can see, the DOS software is probably the most important part of the floppy-disk system. It acts as an extension of the ROM, in that it provides routines for all the various disk operations.

Some of you may be wondering, "How can a piece of software tie in with ROM, which cannot be physically modified?" The answer is that many of the ROM routines have a *detour* through some portion of accessible RAM memory. At certain points in these routines, execution "jumps out" to a RAM address, where the system or operator could place any type of executable program.

In the non-disk system, these addresses (referred to as *DOS links* or *DOS exits*) are coded to redirect execution back to the ROM, resulting in a brief, unproductive loop out of ROM and back again. But we could replace that return instruction with a short routine that would then be executed, and we would have the option of not returning to the ROM afterwards. This is the basic principle of the DOS extension to the ROM.

In similar fashion, there are sets of addresses connected with the exclusive command keywords for the disk version of Basic. The Basic interpreter (in ROM) detects all Basic command words, most of which have associated execution code burned into the ROM. For the Disk Basic words, however, the ROM loops out into RAM for these routines. In non-disk systems, these addresses cause an immediate return to a ROM routine that displays the cryptic message "L3 Error" (or "Level III Error"—the authors of the ROM apparently referred to Disk Basic as Level III Basic). Once again, however, when a DOS is present, these keywords become valid, active commands because the *calling addresses* are redirected to executable routines for the disk system.

The exception to this procedure is the TRS-80 Model II. This computer has essentially no operating code in ROM at all, except for a small initialization routine. What this does is to immediately fetch, from the disk drive, the whole operating system contained on the DOS disk, load it into memory and begin execution once you turn on the computer. The Model II DOS is, therefore, much more sophisticated, as it must include all of the fundamental routines that would otherwise have been in ROM, such as the Basic interpreter, keyboard and video display routines, and so forth. The big advantage is that the user is no longer constrained by a strict ROM code or a given language interpreter—the door has been opened to the use of powerful, esoteric operating systems and other high-level computer languages (Fortran, Cobol, Pilot, CP/M and so on).

A description of DOS features would quickly turn this article into a book, so I'll simply recommend that you go to the people who wrote the book—Radio Shack. Their operating manual for the TRSDOS operating

system is first rate, extremely thorough, and (for the most part) understandable even by the rank amateur. The manual alone is available, or you can buy the entire package including the DOS disk.

TRSDOS itself is fairly simple to learn, and many of the competitive DOS packages are based on Radio Shack's. The other systems generally boast correction of bugs (errors) in TRSDOS, or special routines that TRSDOS doesn't include. At any rate, cut your teeth with TRSDOS and get a good idea of what the DOS is and does before buying one of the new-fangled, high-performance systems.

The market for Model I and III DOS packages is loaded with products, and if you ask 10 different users about their favorite DOS, you are likely to get 10 different answers. Some of the more well known names are Apparat's NEWDOS (there are several versions), Logical Systems' LDOS (formerly VTOS), Micro Systems Software's DOSPLUS, Level IV Products' ULTRADOS, and Cosmopolitan Electronics' MULTIDOS. Newer versions of these products are usually double-density compatible, which is an important factor in Model I and III software interchangeability, and permit conversion of Model I systems to double density.

Before purchasing one of these products, find area computerists who use these systems and see if you can get some hands-on experience with them. Try not to buy on verbal recommendations alone. Also, steer clear of unauthorized copies of a DOS. You can never be sure what condition it is in, or whether it has been periodically upgraded or debugged. Also, you will not get the benefit of follow-up support by the author; most companies distribute upgrades or zaps (recommended corrections) to registered owners only. Furthermore, there is the very real problem of software piracy, and the deprivation of royalties due to software authors. Granted, the price tags on original software often run high, but you will get continuing technical support from the authors and make it financially feasible for them to continue that support and turn out new material. ■

Michael Morra is employed by Enthone Inc. of West Haven, CT. He can be reached at 450 Villa Ave., Fairfield, CT 06432.

Radio Shack: *TRSDOS and Disk Basic Reference Manual* (Version 2.3) Computer Catalog RSC-6

Apparat, Inc.: Documentation for NEWDOS+ and NEWDOS80 (Versions 1.0 and 2.0)

FD 1771-01 Floppy Disk Controller Data Sheet and Application Notes

Shugart Associates: SA400L OEM Manual

IJG Computer Services: *Microsoft Basic Decoded and Other Mysteries*, James Farvour

IJG Computer Services: *TRS-80 Disk and Other Mysteries*, H.C. Pennington

Softside Publications: *Pathways Through the ROM*

80-Northwest Publishing: William Barden, Jr., *TRS-80 Model I Disk Interfacing Guide*

Roxton Baker/The Alternate Source: Documentation for Trakcess

References

See List of Advertisers on Page 563

Binary Breakfast

by Richard E. Esposito

Now you can understand the language of your Model I, III, or Color Computer—it's simple.

All digital computers are based on the binary number system; the TRS-80 is no exception. The binary system is based on the number two; the decimal system is based on 10.

Consider 1,458; in base 10, it represents $1 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 8$, while the number 1011 in base two represents $1 \times 2^3 + 0 \times 2^2 + 1 \times - + 1$.

In base 10, 10 represents $1 \times 10 + 0 = 10$. Therefore, we need symbols that can occupy the right position to represent zero through nine. The symbols we use are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

In base two, 10 represents $1 \times 2 + 0 = 2$, so in base two arithmetic, we need digits representing only zero and one. It is useful to count in both bases (see Table 1).

One of the niceties of dealing with base-two numbers is there are only four addition facts:

$$\begin{array}{l} 1 + 1 = 10 \\ 1 + 0 = 1 \\ 0 + 1 = 1 \\ 0 + 0 = 0 \end{array}$$

Addition in base two is carried out in much the same way as addition in base 10. To add 10101 to 11011:

$$\begin{array}{cccccc} & (1) & (1) & (1) & (1) & \\ & 1 & 1 & 0 & 1 & 1 \\ & 1 & 0 & 1 & 0 & 1 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 \end{array}$$

(d) = digits carried

The multiplication facts are:

$$\begin{array}{l} 1 \times 1 = 1 \\ 1 \times 0 = 0 \\ 0 \times 1 = 0 \\ 0 \times 0 = 0 \end{array}$$

To multiply 101 by 111:

$$\begin{array}{r} 111 \\ \times\ 101 \\ \hline 111 \\ 000 \\ 111 \\ \hline 100011 \end{array}$$

An integer word in the TRS-80 consists of two 8-bit integer bytes, a word being the unit of memory used to store a number.

Let's calculate the binary equivalent of 56. To do this, we use what is commonly known as the Chinese remainder theorem. The process is simple: Divide 56 by two; divide the quotient by two; repeat the process until a quotient of zero is obtained. For example:

$$\begin{array}{rl} 28 & \\ 2\,/\overline{56} & \\ \underline{56} & \\ 0 & \text{first remainder} \\ 14 & \\ 2\,/\overline{28} & \\ \underline{28} & \\ 0 & \text{second remainder} \\ 7 & \\ 2\,/\overline{14} & \\ \underline{14} & \\ 0 & \text{third remainder} \\ 3 & \\ 2\,/\overline{7} & \\ \underline{6} & \\ 1 & \text{fourth remainder} \\ 1 & \\ 2\,/\overline{3} & \\ \underline{2} & \\ 1 & \text{fifth remainder} \\ 0 & \\ 2\,/\overline{1} & \\ \underline{0} & \\ 1 & \text{sixth remainder} \end{array}$$

Since the sixth quotient is zero, we can terminate our division process.

The next step is to assemble our binary number. This is done by writing our remainders in the opposite order of their calculation. That is, starting from the left using 56 as the example: remainder six, remainder five, remainder four and so on, yielding 1 1 1 0 0 0, which is the binary equivalent of 56.

You can double check by doing this: $1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2 + 0 = 56$.

The TRS-80 uses a 16-bit word for integers, so we will add enough zeros on the left to give us a 16-bit number. This yields:

0000000000111000

Now that we have the binary equiv-

Base 10	1	2	3	4	5	6	7	8	9	10
Base 2	1	10	11	100	101	110	111	1000	1001	1010

Table 1

PANDEMONIUM

By Norman J. Wazaney Jr.

alent of 56, how does this relate to the statement: 10 X% = 56? If you follow this statement with: **20 PRINT VAR PTR (X%)**, the computer will print the address of the first byte of the variable X%.

If you add the statements

```
30 PRINT PEEK (VARPTR (X%))
40 PRINT PEEK (VARPTR (X%) + 1)
```

you will get the results 56 and zero from the two locations. How does this compare with our computed binary number? Well, a 16-bit number must be broken into two 8-bit bytes. Doing this with 56 gives us 0 0 0 0 0 0 0 0 and 0 0 1 1 1 0 0 0. The byte on the left is called the high-order byte and the byte on the right is called the low-order byte.

The TRS-80 stores the high-order byte in the second byte of the word and the low-order byte in the first. If you convert the binary numbers from each of these bytes to base 10, you will come up with zero and 56, the same values that were PEEKed from memory.

If you understand everything up to this point, you should be able to explain why the following program yields the result of 32767:

```
10 X% = -1
20 POKE VARPTR (X%), 255
30 POKE VARPTR (X%) + 1, 127
40 PRINT X%
50 STOP
60 END
```

What about negatives?

If you read the *Level II Basic Reference Manual*, you will notice an integer variable may only be as large as 32767. If it were not for negatives, a 16-bit number could go as high as 65535.

The TRS-80 uses a scheme known as two's complement to represent negative integer numbers. What this scheme does is use half the possible 65536 values to represent positive numbers and the other half for negatives.

So how do we get −56? Recall that 56 was equal to 0000000000111000. To compute −56, first calculate the one's complement by switching all ones to zeros and all zeros to ones. This gives us 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1. To the one's complement add one to come up with the two's complement. In the case of −56, this yields:

```
  1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1
+                               1
  -------------------------------
  1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 0
```

If this addition had resulted in a 17th digit for our result, it would have been discarded. Now as before with 56, we split the result into two bytes, this time resulting in 1 1 1 1 1 1 1 1 and 1 1 0 0 1 0 0 0. Converting these bytes to base 10 results in 255 and 200.

You are now equipped to explain why this program yields −56:

```
10 X% = -1
20 POKE VARPTR (X%), 200
30 POKE VARPTR (X%) + 1, 255
40 PRINT X%
50 STOP
60 END
```

Floating Points

How does the TRS-80 handle floating-point numbers? To answer that question we must analyze the binary number system still further. In binary:

$$0.1 = 1/2$$
$$0.01 = 1/4$$
$$0.001 = 1/8$$
$$0.0001 = 1/16...$$

To convert fractions to binary we multiply the decimal fraction by two. If the result is greater than zero, write down the number or else write down a zero.

Continue multiplying only the frac-

tional part of the product until you have accumulated the proper number of bits. In the case of 56.1625, 56 gave us six bits but a single-precision variable in Level II Basic requires 24 bits in addition to the exponent, so we proceed as follows:

	.1625
	×2
Bit 1	0.325
	×2
Bit 2	0.65
	×2
Bit 3	1.3
	×2
Bit 4	0.6
	×2
Bit 5	1.2
	×2
Bit 6	0.4
	×2
Bit 7	0.8
	×2
Bit 8	1.6
	×2
Bit 9	1.2
	×2
Bit 10	0.4
	×2
Bit 11	0.8
	×2
Bit 12	1.6
	×2
Bit 13	1.2
	×2
Bit 14	0.4
	×2
Bit 15	0.8
	×2
Bit 16	1.6
	×2
Bit 17	1.2
	×2
Bit 18	0.4

We now have 18 additional bits in 56.1625 = 111000.001010011001100110. Now we normalize the number by moving the binary point (binary point is to the binary system as decimal point is to the decimal system) to the leftmost position and put the number into scientific notation. The result is a binary point, followed by a one, followed by additional bits. If the original number were a fraction, the binary point would be adjusted to the right to achieve this status. (You may also have to calculate more bits to achieve the required precision: 24 bits to the right of the binary point.) This yields 56.1625 = .11100 0001010011001100110 $\times 2^6$. Now break the number into bytes as we did with the integer variable. This yields: most significant byte = 11100000; next most significant byte = 10100110; and least significant byte = 01100110.

Now take the power of two and add it to 128. In this case, we get 134. For a positive value, change the leftmost bit of the most significant byte (the byte closest to the binary point) from a one to a zero. For a negative value, omit this step. This is the system adopted by Microsoft for representations of positive floating-point numbers. It is by no means universal. In the case of 56.1625, we have 01100000.

Next convert your three significant bytes to base 10. We now have 96, 166 and 102. The number 56.1625 would be stored in the TRS-80 using four consecutive bytes:

Byte 1 (least significant byte) = 102
Byte 2 (next most significant byte) = 166
Byte 3 (most significant byte) = 96
Byte 4 (exponent) = 134

If the following program is run on the TRS-80, the printout would yield

✓See List of Advertisers on Page 563

56.1625:

```
10 X = -1
20 V = VARPTR (X)
30 POKE V,102
40 POKE V+1,166
50 POKE V+2,96
60 POKE V+3,134
70 PRINT X
80 STOP
90 END
```

For TRS-80 double precision, the process is similar, except you need 56 bits to the right of the binary point in the normalized number.

In the case of 56.1625: 56.1625 = 111000.00101001100110011001100110 0110011001100110011001100110. After changing to scientific notation, we have: .11100000101001100110011001100 1100110011001100110011001100110 $\times$ 2^6. The bytes going from most significant to least significant and converting to base 10 are:

```
01100000 = 96 (the left bit was stripped as
              before.)
10100110 = 166
01100110 = 102
01100110 = 102
01100110 = 102
01100110 = 102
01100110 = 102
```

and, as before, the exponent is 128 + 6 = 134.

The number is stored in very much the same way as in single precision except it uses 8 bytes. If we run the following program, the result would be 56.1625:

```
10 X# = -1
20 V = VARPTR (X#)
30 POKE V, 102
40 POKE V+1,102
50 POKE V+2,102
60 POKE V+3,102
70 POKE V+4,102
80 POKE V+5,166
90 POKE V+6,96
100 POKE V+7,134
110 PRINT X#
120 STOP
130 END
```

For the TRS-80 Color Computer, let's again consider the number 56.1625. The Color Computer only has one type of variable, which is a five-byte, floating-point number.

As before, convert 56.1625 to binary, this time to 32 bits: 56.1625 = 111000.00101001100110011001100110. After changing to scientific notation we have: .11100000101001100110011001100 11001100110 $\times$ 2^6. Break into bytes and convert to base 10:

```
01100000 = 96 (the left byte was stripped as
              before)
10100110 = 166
01100110 = 102
01100110 = 102
```

and as before, the exponent is 128 + 6 = 134.

Unlike before, the numbers are stored in the reverse order of those in the Models I and II. This order is used in most computers. The real oddball in this case is the Z80. The TRS-80 uses Motorola's 6809 chip.

As before, if we run the following program, 56.1625 will be our result:

```
10 X = -1
20 V = VARPTR (X)
30 POKE V,134
40 POKE V+1,96
50 POKE V+2,166
60 POKE V+3,102
70 POKE V+4,102
80 PRINT X
90 STOP
100 END
```

The following program written in TRS-80 Basic divides an input number by two by changing the value of the exponent of the stored floating-point number:

```
10 INPUT X
20 V = VARPTR (X)
30 E = PEEK (V+3)
40 E = E-1
50 POKE V+3,E
60 PRINT X
70 STOP
80 END
```

The corresponding program for the Color Computer follows:

```
10 INPUT X
20 V = VARPTR (X)
30 E = PEEK (V)
40 E = E-1
50 POKE V,E
60 PRINT X
70 STOP
80 END
```

Now consider this problem raised by Michael Binkhurst ("80 Aid," *80 Micro*, February 1981): "When the following program is run, why is the result .0100002 instead of the expected .01?"

```
120 A = 20.01:B = 20:PRINT A-B
```

To see why, convert both numbers to binary as the TRS-80 does.

20 = 10100
.01 = .0000001010001111010

(19 places are required in this problem)

In scientific notation:

20.01 = .1010000000001010001111010 $\times$ 2^5
20 = .101000000000000000000000 $\times$ 2^5

If we subtract the two, we obtain:

.01 ?= .0000000000001010001111010 $\times$ 2^5

This result must now be normalized resulting in:

.01 ?= .101000111101000000000000 $\times$ 2^{-6}

(the underlined zeros were added to retain 24 bits of accuracy)

Converting our result back to base 10 yields: $(1/2 + 1/8 + 1/128 + 1/256 + 1/512 + 1/1024 + 1/4096) \times 2^{-6} = .0100002$

The error introduced into this calculation plagues computers that use scientific notation. One way to avoid this problem is to use the binary-coded decimal arithmetic scheme where each decimal digit is represented by four bits and the computer then calculates in much the same manner as one would using pencil and paper. This technique is fine for business problems, but it uses more memory and is slower.

In the world of big computers, both systems are used: Fortran (with floating-point arithmetic) for scientific problems and Cobol (with binary-coded decimal arithmetic) for business problems.

Some possible solutions to this problem include:

- Use double precision. It will not eliminate the error completely, but it will be made smaller.
- Get a computer that uses BCD arithmetic. An example of one is the APF Imagination Machine.
- Use the technique called "fuzzing." If a number is within a certain tolerance of a base-10 number, print that base-10 number. This scheme is used in APL. In APL, the "fuzz" is adjustable. ■

Richard Esposito can be reached at 62 Meadow Road, Frostburg, MD 21532.

Announcing the best Error Free Personal Computer Diskette Money can Buy. For Less.

- **Error Free**
- 1 year warranty
- Hub ring installed
- Write/Protect notch
- Next day delivery

$19.90/box of 10

- **No** minimum order quantity
- 8" or 5¼"
- Plus 2 Free BONUS DISKS/Box*

*5¼" SPECIAL ONLY

If you are a member of a user group or a school district please call for special terms on future offers. TSS is the largest specialty supplier of magnetic media in the Midwest. We have the products that you want when you need them. Please take advantage of this introductory offer and call us **now.**

TSS Transaction Storage Systems, Inc.
MAGNETIC MEDIA SPECIALISTS

22255 Greenfield Rd.
Southfield, Michigan 48075

44

CALL TOLL FREE
1-800-521-5700
1-800-482-4770 (Michigan)
313-557-3036 (Detroit)
312-922-0076 (Chicago)
614-221-1788 (Columbus)
513-621-1518 (Cincinnati)
Telex 810-224-4646

EXPECT A MIRACLE

YES, TSS is the magnetic media supplier that I have always wanted.

Please send me ______ (Box Quantity), 8"☐ 5¼"☐, @ ______ (Price) ea.

I am interested. Please send me more information or call me at (   ) ______

For faster order entry call any of our toll-free or local numbers

Company ______

Name ______ Title ______

Address ______

City ______ State ______ Zip ______

Amex/Master Card/Visa orders are accepted — *Expiration Date*

__ __ __   __ __ __ __   __ __ __ __ __   *month ____ year ____*

See List of Advertisers on Page 563

Nine Programming Tricks

by David D. Busch

Try these shortcuts for faster Basic programming—easier backups, macro commands, renumberings, cross references, and block moves.

Many programmers do not use all the tools available to them as they write their code. When a programmer lets his machines handle as much of the drudgery as possible, he can then devote a full effort to the creative aspects of programming.

Leading software suppliers have helped by offering utilities, disk operating systems with enhanced Disk Basics, and other tools to make programming easier. Even so, while the documentation for these time-savers is usually adequate, it lacks real-life examples on how to use them to optimize your programming productivity. I will present nine shortcuts that make my programming easier.

Easier Backups

As you develop a program, it is good practice to periodically save the work in progress either to cassette or disk. Thus, should a power failure occur, hours of work are not lost. With a disk-based system, making backups is easy—so simple that when I end a session and look at the working disk's directory, I see 10 or more versions of my program tucked away (PROGRAM/V1, PROGRAM/V2 and so on). This system works well, but when saving, I can never remember what I called the last version. I must either invoke a CMD"DIR" to check, or play it safe and skip a number or two.

Here is a short program disk users can append onto the end of any program they are working on and use to automatically save an updated version of a program under an appropriate file name. To use it, remember to turn on your clock. Then, by typing GOTO 30000 at any point during program development, the module will collect the current TIME$, extract the hour and minutes, and use that to make the file name.

```
30000 B$=TIME$:H$=MID$(B$,10,2)
30010 M$=MID$(B$,13,2)
30020 F$="PROG"+H$+M$
30030 SAVE F$
```

Save this in ASCII form on your disk, and then append or merge it to any program you choose (which does not have line numbers that conflict). You may want to edit line 30020, replacing the string "PROG" with any four letters that are more meaningful for the program you are developing. If you want to back up the program to two (or more) disk drives automatically, add the following lines:

```
30025 F1$=F$+":1":F2$=F$+":2"
30035 SAVE F1$:SAVE F2$
```

This automatic save feature provides you with valuable added security.

Macro Commands

Computer users who have an operating system or utility (such as IRV) that provides a keystroke multiplier or defined key functions can speed up their program development even more. You can store the short program above in one of IRV's defined keys and invoke it by hitting one or two keys. Other frequently used commands, such as CMD"DIR :1" or RENUM 10,10, can also be applied to user-programmable keys.

Though programmers most frequently use such utilities to speed the entry of keywords (I instead of Input), you can define and reuse longer, more complex strings. How often have you typed the following:

```
A$=INKEY$:IF A$="" GOTO
OPEN "O",1,F$
HIT ANY KEY TO CONTINUE
CLS:PRINT:PRINT
```

Defining these will save you many more keystrokes than simply bypassing Input with I. Try it.

Moving Blocks of Code

Renumbering programs is a basic programming tool. Even if you code with line numbers in increments of 50, some day you will have to fit a line between line 350 and line 349. Renumbering utilities such as NEWDOS80 2.0's RENUM make this problem trivial. Almost as quickly as you can type RENUM 10,10, (or RENUM ,) the entire program will be renumbered with plenty of room to add new lines.

But wait a minute! While writing the code, don't you keep many line numbers in your head? It is handy not to have to search for your main menu routine and just type GOTO 200 at the end of each module. If you renumber frequently, a module that was at line 200 might have been moved back to line 180, or ahead to line 220. You were careful to put the subroutines at lines 2000, 3000, 4000 and 5000 so you could find them. Now they have been renumbered to lines 1340, 1350, 1360 and 1370. NEWDOS80 2.0's RENUM can be very flexible. Try these formats, all carefully explained in the document-

ation, but not often used. If necessary, write them on a sticker and place it on your computer for reference. These formats save time: RENUM first line, increment, first line renumbered, last line renumbered.

In this example, first line should be the beginning line number for the renumbered section. Increment should be the amount of space desired between lines, while the last two entries are the first line to be renumbered and the last to be renumbered. This format allows you to renumber only *part* of a program, without interfering with the rest of the code.

Suppose you are running out of space between lines 300 and 400, but have no additional program lines written between lines 400 and 500. You can expand the code-filling lines 300–400 to include all numbers between lines 300 and 500.

RENUM 300,10,300,500 accomplishes this. You can use two commas to replace the 10, a default value, so that RENUM 300,,300,500 accomplishes the same thing. All program lines beginning at line 300 and ending at line 500 will be renumbered with an increment of 10. The first line will be assigned 300, the second 310, and so forth. The highest-numbered line of the old module, line 400, will now have a higher number past 400, possibly as high as 500. The other lines will be separated by 10 lines in the code. The rest of your program remains unchanged.

Have you written a title block, remarks, or some other piece of code at the beginning of a program that you do not want to renumber? In this case, you want to renumber the program from line X onward. Use this format: RENUM first line number, increment, first line to be renumbered.

If I wanted to renumber all program lines from 10 onward with an increment of 5 and use line 25 as the first line number for that module, I would enter RENUM 25,5,10.

Because there is no fourth entry (last line number), the utility renumbers all remaining lines in the program.

How about moving blocks of program lines? NEWDOS80 2.0 has a quick and dirty DI and DU command that moves single lines fast (or duplicates them in a new location). Unfortunately, this feature does not change any GOTOs that might be affected by the change. If your program contains a branch to a line that was moved, you must find the line and make the correct change. This is not difficult (using REF, discussed later), but is an additional opportunity for bugs to creep in.

However, RENUM can be a better way of moving whole blocks of code, because all GOTOs and GOSUBs are updated to take into account the change. Although the syntax is identical, I have phrased it slightly differently in the following example to make it clearer exactly what is taking place: RENUM new location, increment, first line moved, last line moved.

This format takes a section of code and renumbers it (if necessary) while dropping it neatly into a new location. As an example, assume we want to move all lines between line 300 and line 400 to a space between line 1000 and line 2000. I will use a line number increment of 10. Type RENUM 1000,10, 300,400.

That's all there is to it. Once you are comfortable moving blocks of code around, you might want to try one other trick: Save modules that are frequently used in ASCII form on your disk. These may be disk I/O routines, menus, title blocks and so forth. Then merge or append them with the program under development, and move

them as you see fit. All the internal GOTOs and GOSUBs within the modules will be renumbered to match the new location. You merely need to add references to transfer control to and from your transplanted subroutines.

RENUM has one other useful programming feature. You may type RENUM U, and the utility will not actually do any renumbering. Instead, it will search through your program and find any undefined line numbers and other errors that would abort a renumber. In this way, you may find and delete redundant blocks of code or errors.

Cross-Reference Utilities

A number of sources have cross-reference utility programs that search your Basic code or find all occurrences of a variable, line number, or text strings. These can be useful debugging tools. Are you interested in finding out why a variable changes value when it is not supposed to? Check out all uses of that variable. You may find A$ was used in an INKEY$ line within a module that also includes a LINEINPUT A$ statement.

Long-time NEWDOS80 1.0 and 2.0 users are not aware of all the things REF can do. Most know that typing REF A produces a listing of all uses of A as a variable, either string, integer, array, double precision or single precision. But then they laboriously type LIST 10, LIST 300, LIST 1000, and so forth to check each use.

This is unnecessary! After invoking the REF variable, simply type REF with no argument. The utility will list the first line where the variable is used. You may edit that line or ignore it as you prefer. Type REF again and the next text line with that variable will be shown. Text End appears when all have been displayed. This is simple, but often overlooked.

REF will also search for character sequences, such as PRINT, HIT ANY KEY, and so forth, up to 16 characters. By prefacing the string with a quotation mark (REF"text string) the program will search strings and comments for the desired characters. This is a fast way to find a certain block of code, or to change all Prints to LPRINTs or do some other work.

These nine programming tricks should be useful to anyone doing more than casual programming. After all, the computer should be doing as much of the work as possible. ■

So Far, So Good...

Your Model III is a fine computer. So why settle for less than a fine disk storage system?

A fast, reliable disk storage system — no other device is so vital to the satisfactory performance of your computer. At Percom we build quality, high-performance disk drive systems. From gold-plated connector contacts to goldcad metal chassis structures. From proven design through 100% reliability testing. Percom disk systems are the standard by which others are judged — the industry's "gold standard," in a sense. And since Percom is the largest independent manufacturer of disk systems for microcomputers, you get Percom quality at very competitive prices. Add-on drives for TRS-80* computers start as low as $399. Complete first-drive systems for the Model III start at only $599. Put a quality Percom mini-disk storage system in your Model III. And save. 1

#1 For Your Model III

Percom TFD floppy drives for your TRS-80* Model III computer are available in single- and dual-head versions, providing 180,000 and 360,000 bytes per drive, respectively, of formatted on-line storage capacity.

The Percom Model III disk controller handles one to four drives so you can have from 180,000 to over 1.4 million bytes of on-line program and file storage.

You get MSS' DOSPLUS‡ with first-drive systems, and your first drive may be either internal (add-in) or external (add-on). Percom TFD drive units also work with TRSDOS* and other compatible disk-operating systems.

First-drive systems are pre-assembled and may be installed with common tools. 2

#1 For Your Model I, Too

Percom TFD add-on drives for the TRS-80* Model I computer are available with the same features and the same quality control measures as TFD Model III drives. As for Model III drives, all Percom Model I drives are double-density rated. Install Percom's popular DOUBLER II in your Model I Expansion Interface and upgrade your Model I to provide the same storage capacity as the Model III. 3

Now! Low-cost Percom Winchester Disk Systems for Radio Shack computers.

To order, or for the name of your authorized Percom retailer, call toll free **1-800-527-1222.**

PERCOM

The Drive People
You get more out of Percom disk systems.
Expect it!

PERCOM DATA COMPANY, INC.
11220 Pagemill Road • Dallas, Texas 75243
(214) 340-7081

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE.

*TRS-80 is a trademark of Tandy Radio Shack Corporation which has no relationship to Percom Data Company, Inc.
PERCOM, DOUBLER II and OS-80/III are trademarks of Percom Data Company, Inc.
‡trademark of Micro Systems Software, Inc.

Yes ... I'd like to know more about the best for my Radio Shack computer. Send me free literature about quality Percom products:

☐ Mdl I ☐ Mdl III
☐ Mdl II ☐ Mdl 16

Send to
PERCOM DATA COMPANY, INC. Dept. 8-M
11220 Pagemill Road, Dallas, TX, 75243

NAME ____________________

STREET ____________________

CITY ____________________ STATE ________

ZIP ____________________ PHONE NUMBER ________

See List of Advertisers on Page 563

The Sum of Its Parts

by Spencer Weersing

Good Basic programming begins with an organized game plan. This article explains how to tackle your programming in a logical manner.

"A confusion of aims and a profusion of means seems to be our main problem."—Albert Einstein

Beginning students of Basic often have difficulty writing programs that work properly because they use poor design techniques. But if you plan your work, then work your plan, you will be successful. Follow these five steps of good design technique; they will lead you to proficiency:

- State the purpose of the program.
- Make an organizational chart.
- Make a pseudocoded outline.
- Code and check the fundamental program.
- Code and check the completed program.

The First Step

Succinctly state the purpose of the program. Be as specific as possible without stating the method by which you plan to accomplish that purpose. You need to be clear, concise, and complete. Ambiguous writing and disorganized thoughts will destroy a program before it is born.

The Key Box

Model I
16K RAM
Cassette Basic
Disk Basic

If there are multiple things that the program must do, divide it into appropriate subprograms and state the purpose of each. (A subprogram performs a major part of the work of the whole program.) If you divide your task into parts, you can write and check manageable portions to see that each part performs its functions properly. The sum of the parts, each fulfilling its purpose, equals the whole. Complex programs are hard to write any other way because, without subprograms, you have to contend with the entire mass of computer code; this defies understanding. Split the program into parts that you can easily handle.

As an example, let's write a program that has practical value when you are considering purchase of real estate. Assume that you want to know if the proposed terms of a land contract are reasonable and within your means. You have to judge if the proposed amount of the total loan, the down payment, the monthly payments, and the interest seem satisfactory, and you want a program that calculates the duration of the contract in order to facilitate that judgment.

For instance, if the down payment or the monthly payments are too low, the monthly interest would not be covered or the contract would run for an excessive number of years. If the contract is not satisfactory with the given terms, how long would it run with other terms? Alternatively, if you know the principal, interest rate, and duration, how much would the monthly payments have to be?

Once all terms seem satisfactory, you want to know four things each month: how much of the current monthly payment covers interest; how much of the current monthly payment reduces the principal; how much the new principal is; and the total to date of interest payments (for tax pur-

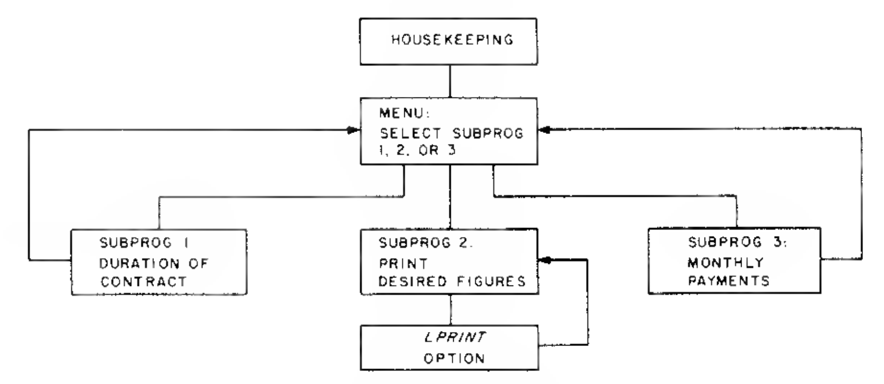


Fig. 1. Organizational Chart

poses). The program, to be useful for others who may not have the same hardware and peripheral equipment that you do, must have an option to print copy if a printer is available.

Now that you have considered your needs, state the purpose of the program: calculate a land contract. First, calculate its duration, given the principal, interest rate, and monthly payments. Second, for each month, display payment portion for interest, payment portion reducing principal, new principal, and total interest to date. Have an option to print a payment schedule on a line printer. Third, calculate monthly payments if the duration is known.

Plan on three subprograms to fulfill your needs.

Step Two

Next, make an organizational chart (see Fig. 1). This is a simple way to visualize the sequence of the major parts of the program. Do not confuse this with a flowchart, a diagram that is helpful to demonstrate how the computer follows a specific instruction. After you have written the program, flowcharts help you understand computer logic, but they are far too detailed at this stage. If you attempt to make a flowchart too early in programming, you become so immersed in the minor details of Basic technique that you lose the overall design. Do not concern yourself with coding techniques at this stage, only with the overall goals of the programs and subprograms. You have to know where you want to go before deciding if it is better to take a train or a bus! The road map that guides you to your destination is the organizational chart.

The first item in the organizational chart is housekeeping, a part that sets things up in preparation for a data-processing operation. The second item in the chart is normally the master program (do not confuse this with the main program itself). The master program directs the proper sequence of subprograms to accomplish the overall purpose. If the sequence of subprograms is under operator control by means of keyboard input, the master program is often called the menu, because the operator makes selections from a list of subprograms. The master program also allows flexibility in the sequence of operations and easily traced program flow.

In the land contract organizational chart, the menu offers a choice of any of the three subprograms. Following subprogram one or three, program flow returns to the menu. Following subprogram two, the user may produce a printout by rerunning subprogram two with the necessary instructions. You can easily visualize program flow by using an organizational chart.

The Third Step

Next, outline each subprogram (see Fig. 2). Write the purpose of each, and list (in appropriate sequence) the data you need to give to the computer (input), what you expect the computer to do with it (process), and what results you expect (output). These are the three parts of a computer program. A parallel outline under these headings is called pseudocode. Pseudocode is not code that the computer understands, but detailed directions that make actual coding obvious.

Input may be accomplished from the keyboard, from a data file on tape or disk, or from previous computer process. The process section is the actual

Subprogram 1: Calculated duration of proposed land contract.

Input	**Process**	**Output**
total amt TA		
down payment D	TA – D	P = amt of contract
% interest I	I/100	decimal form
monthly payment MP		
month X = 0	X + 1	X = next month
	P*I/12	MI = amt for interest
	IF MP<MI1	Stop
	MP – MI	MR = amt applied to P
	P – MR	P = new principal
	IF P≤0	Print X (= # months)
		Restore input
	GOTO start	GOTO Menu

Subprogram 2: For each month (with option to print via line printer) display month, payment portion for interest, payment portion reducing principal, new principal, and total interest to date.

Input	**Process**	**Output**
same	X + 1	X = total # months
	P*I/12	MI = amt for interest
	MP – MI	MR = amt applied to P
	P – MR	P = new principal
	MT + MI	MT = total I to date
		Print X MI MR P MT
	IF P≤MP	Print "Pay" P + P*I/12
		Total = MT + P + P*I/12
	GOTO start	GOTO LPRINT option
	make flag to LPRINT via subroutines	
	restore input	
	GOTO start	

Notes:
Make P and MT double-precision values if over 6 significant figures; store them without trailing garbage digits. Precede Menu with input that is common to all subprograms. Include line to restore X, MT, and P.

Subprogram 3: Calculate monthly payment if given term.

Input	**Process**	**Output**
same		
term (years) N		
	I*100	% form
	P*I/1200	Y = intermediate for MP
	(1 + I/1200)↑(12*N)	Z = intermediate for MP
	Y/(1 – 1/Z)	MP = monthly payment
	I/100	decimal form
	GOTO Menu	

Note: Round off MP to $.01.

Fig. 2. Outline

tions, separate each part widely and use an interval of 10 between line numbers. You can renumber the program later, if you desire.

Begin the housekeeping section with a remark statement giving a proper file name, current date, and the author's name. This important information allows proper magnetic storage of the program in a file, identifies the file later, and tells the user which modification the program is. You are limited to eight alphanumeric characters for the file name, and the first character must be alphabetic. Truncate the program title "Land Contract Schedule" to LANDCONT for an acceptable file name that helps you remember which program it is. Radio Shack's Level II Basic allows you to put an extension of three characters following a slash on the file name to identify the data the file contains. Use /BAS because this file contains a program written in Basic. The author's name is important for proper credit—or to assign responsibility if the program fails to operate properly!

Add additional remark statements throughout the program as needed for clarity. While this consumes memory and makes the program run more slowly, remark statements help recall essential information about any particular operation that you are instructing the computer to do. Once the program is perfected you can delete remark statements, but keep an annotated printout for your reference library.

Set aside housekeeping line numbers 500–799 for subroutines. A subroutine is a set of instructions that will be needed more than once. Rather than requiring you to repeat the same instructions, Basic allows access to those instructions via the subroutine function. After performing the work directed by the instructions, program flow returns to the location of origin in the main program. You can use one set of instructions multiple times from any part of the program (even from another subroutine). Include a remark statement with each subroutine indicating which lines call it, so you may easily trace program flow.

Line 500 restores original input values to allow succession of subprograms without having to repeat input. Lines 600 and 610 are reserved for the print option in the completed program.

Do not use any housekeeping functions in the fundamental coding that are not absolutely required. Discipline yourself to include essentials only, regardless of how desirable nonessentials are. You can add elegance after the program functions perfectly. (Modify it to run faster, or produce an ideal display or attractive printout, for example.)

Lines 900–940 are not part of housekeeping. Some identical keyboard input is required by all of the subprograms, so this input precedes the subprograms.

The next section, lines 1000–1040, is the menu, which allows a choice of any subprogram.

Now, code each subprogram. Once again keep your priorities clear: Make the program run correctly before worrying about attractive output, speed running, and nonessential process. As you work, update the list of variables and their purposes. You must know if you have used a variable previously, and if you have defined it in the housekeeping section in a manner that currently makes it an inappropriate choice. (If you want to use a single-precision variable, you must know if you have previously defined it as an integer or a string variable.)

Subprogram 3 is based on the following formula:

$$M = \frac{PI/1200}{1 - 1/(1 + I/1200)^{12N}}$$

in which M is the monthly payment, P is the principal of the land contract, I is the interest rate in percent, and N is the length of the mortgage in years.

Accustom yourself to writing and reading code without any spaces between characters. Spaces waste memory and room on the video screen. Youcangetusedtoreadingpackedprogramsandtheincreasedefficiencyis worthit! Avoid multiple statement lines during the first coding because they make additions and editing difficult. Also, if you ever find yourself coding a repetitive, endless series of steps, stop! Get advice from a more experienced Basic programmer who may know a way to accomplish the task more efficiently.

Debug each subprogram as it is written. Run it with appropriate input and be sure the output is exactly correct (see Fig. 3). Take nothing for granted. Computer outputs look so credible that it is easy to believe that the results must surely be correct. However, computers make mistakes more quickly than any other human contrivance! Use the highest and lowest expected values to be sure the program can handle them. Put the program through every imaginable test. At this stage you will appreciate having kept everything as simple as possible. After the fundamental program runs satisfactorily, be assured that you have accomplished the most difficult part of your task.

Last Step

The final step for good program design is to code and debug the completed program (see Program Listing 2). Modify the fundamental program to decrease the running time, increase accuracy, improve the video display and line-printer output, and make the final result exactly what you stated in step one. Carefully debug each feature you add as it is written, otherwise a formidable task lies ahead!

I expanded the housekeeping section from the lone, but important, remark statement of the fundamental program. CLEAR 100 reserves 100 bytes

WEERSING/GREEN Land Contract
25500 less 5000 down = 20500 initial principal at 11% interest
Payment of 350 due each month on 7th

Month	Monthly interest	Principal reduced	New principal	Interest to date
1	187.92	162.08	20337.92	187.92
2	186.43	163.57	20174.35	374.35
3	184.93	165.07	20009.28	559.28
4	183.42	166.58	19842.70	742.70
5	181.89	168.11	19674.59	924.59
•	•	•	•	•
79	19.75	330.25	1824.18	8974.18
80	16.72	333.28	1490.90	8990.90
81	13.67	336.33	1154.57	9004.57
82	10.58	339.42	815.15	9015.15
83	7.47	342.53	742.62	9022.62
84	4.33	345.67	126.95	9026.95

Final payment of 128.11 next month
Total interest on land contract will = 9028.11

Fig. 4. Final Payment Schedule

computation that is required for each step in every subprogram. The output may be the desired end product, or it may be data that will be input once more for further computer process.

Study Fig. 2, which shows pseudocoding of the land contract program. Particularly note these important features:

- The pseudocoding for each subprogram is short. You have to consider only a small portion at a time.
- The format of the outline leads you to work as a computer does: input, process, and output.
- Each variable is defined accurately when first introduced. Keep a separate list of variables (Fig. 5) to avoid confusion when you later assign variable names in other subprograms.
- Any helpful notes are included at the bottom of each subprogram.
- The outline contains only the essentials; all unnecessary features are left out. This helps prevent errors that are difficult to find and correct.

Coding this plan will be easy, checking it will be easy. After the initially coded program is functioning perfectly, you will have ample opportunity to flesh out the program to make it run more quickly and accurately, and to produce an attractive output on screen and paper.

Prior to the actual coding, rewrite any of the preceding three steps you think necessary. This is important. The extra time you take to think through each subprogram will be amply repaid when coding and debugging your program. (Debugging a program means finding and fixing any errors in it. Debugging a program can often take longer than writing it!) Do not hesitate to rewrite and edit your work if you can think of any improvements.

Debugging can be relatively painless if you split programs into subprograms, with each procedure of each subprogram divided into the three components of computer process: input, process, and output. Keep your outlines as simple as possible. Fancy output should follow the development of a working program, not precede it.

```
INPUT TOTAL AMOUNT? 25500
INPUT DOWN PAYMENT? 5000
INPUT % INTEREST? 11
INPUT MONTHLY PAYMENT? 350
INPUT 1 TO FIGURE NUMBER MONTHS
INPUT 2 TO FIGURE DETAILS AND PRINT
INPUT 3 TO FIGURE MONTHLY PAYMENT
?1
 85 MONTHS TO RUN
INPUT 1 TO FIGURE NUMBER MONTHS
INPUT 2 TO FIGURE DETAILS AND PRINT
INPUT 3 TO FIGURE MONTHLY PAYMENT
?2

       1      187.917     162.083     20337.9     187.917
       2      186.431     163.569     20174.3     374.348
       3      184.932     165.068     20009.3     559.279
       4      183.418     166.582     19842.7     742.698
       5      181.891     168.109     19674.6     924.589
       6       180.35      169.65     19504.9     1104.94
       •         •           •           •           •
      79      19.7488     330.251     1824.16     8974.17
      80      16.7215     333.279     1490.88     8990.89
      81       3.6664     336.334     1154.55     9004.55
      82      10.5834     339.417     815.135     9015.14
      83      7.47207     342.528     472.607     9022.61
      84      4.33223     345.668     126.939     9026.94
PAY 128.102 NEXT MONTH; TOTAL INTEREST TO DATE WILL  =9037.52
```

Fig. 3. Run of Fundamental Programs

Step Four

After you are satisfied with your preceding work, write the fundamental program in Basic (see Program Listing 1). Assign line numbers 1–999 for housekeeping, 1000–1999 for the menu, 2000–2999 for subprogram one, 3000–3999 for subprogram two, 4000–4999 for the option to print, and 5000 on for subprogram three. To accommodate later refinements and addi-

See List of Advertisers on Page 563

for string use. (Fifty bytes are automatically reserved, but more are needed because of line 520.) All numeric variables beginning with X are defined as integers. C$ and M$ are field specifiers for Print Using statements. Your program may also need housekeeping chores such as dimensioning arrays or defining integral, single, and double precision of other variables. In long programs, define integers for use in For . . . Next loops for faster program execution, unless you require greater precision. Strings may also need to be defined.

You may need to define commonly used values with variables (instead of repeatedly using constants), because it takes less time to access a variable than to convert a constant to floating point representation. Define the most commonly used variables first so they are at the top of the variable table list, because the computer searches the table from top to bottom to find the variable.

You may need to initiate an On Error GOTO statement, or to code Read . . . Data statements, or to enable the use of machine-language subroutines through the DEFUSR function.

Use the list of variables (see Fig. 5) to

Program Listing 1. Fundamental Program

```
10 'LANDCONT/BAS, 11/10/81, by Spencer Weersing. Land Contracts
500 GOTO900'     *      *      *      *      *  Subroutines 500-899
510 P=OP:MT=0:X=0:RETURN'        restores input 2060 4010 5060
600 RETURN'                                  future LPRINT 3070
610 RETURN'                                  future LPRINT 3080
900 '      *      *      *      *      *      input for subprograms
910 INPUT"Input total amount";TA
920 INPUT"Input down payment";D:P=TA-D:OP=P'      orig principal
930 INPUT"Input % interest";I:I=I/100'              decimal form
940 INPUT"Input monthly payment";MP
1000 '     *      *      *      *      *      *      *      menu
1010 PRINT"Input 1 to figure number months"
1020 PRINT"Input 2 to figure details and print"
1030 PRINT"Input 3 to figure monthly payment"
1040 INPUTX1:ONX1GOTO2000,3000,5000
2000 '     *      *      *      *      *      *      *  subprogram 1
2010 X=X+1'                                      number of month
2020 MI=P*I/12'                                 amt for interest
2030 IFMP<MIPRINT"Won't work!":STOP
2040 MR=MP-MI'                           amt reducing principal
2050 P=P-MR'                                       new principal
2060 IFP<=0PRINTX"months to run":GOSUB510:GOTO1000
2070 GOTO2010
3000 '     *      *      *      *      *      *      *  subprogram 2
3010 X=X+1
3020 MI=P*I/12
3030 MR=MP-MI
3040 P=P-MR
3060 MT=MT+MI'                           total interest to date
3070 PRINTX;MI,MR,P,MT:IFFL=1GOSUB600'   FLag for LPRINT option
3080 IFP<=MPPRINT"Pay"P+P*I/12"next month; total interest to dat
e will ="MT+P*I/12:IFFL=1GOSUB610:ENDELSE4000
3090 GOTO3010
4000 '     *      *      *      *      *      *      *  LPRINT option
4010 INPUT"Print, Y or N";X$:IFX$="Y"THENFL=1:GOSUB510:GOTO3000
4020 END
5000 '     *      *      *      *      *      *      *   subprogram 3
```

Listing 1 continues

Listing 1 continued

```
5010 INPUT"Input term in years";N:X2=N*12'        X2=term in mos
5020 I=I*100'                                              % form
5030 Y=P*I/1200:Z=(1+I/1200)[(12*N):MP=Y/(1-1/Z)'     MP formula
5040 PRINT"Monthly payments =";:PRINTMP
5050 MP=INT(MP*100+.5)/100'                          round to .01
5060 I=I/100:GOSUB510:GOTO1000'decimal form, restore, goto menu
```

Program Listing 2. Completed Program

```
10 'LANDCONT/BAS, 11/30/81, by Spencer Weersing. Land contracts
100 '     *     *     *     *     *     *     *     housekeeping
110 CLEAR100'                                  bytes for strings
120 DEFINTX:C$="#######.##":M$="###"'         format LPRINTUSING
130 CLS:PRINT"Land Contract Schedule":GOSUB520
500 GOTO900'   *     *     *     *     *     *       subroutines
510 P=IP:QT=0:X=0:F1=0:RETURN'    restores input 2060 4010 5060
520 PRINTSTRING$(64,"-");:RETURN'                       130 3070
530 LPRINTSTRING$(43,"-"):LPRINT:RETURN'                 640 660
540 T#=P:DEFDBLP:P=T#:RETURN'        converts P to dbl prec 3016
550 T!=P:DEFSNGP:P=T!:RETURN'        converts P to sng prec 3018
560 T#=QT:DEFDBLQ:QT=T#:RETURN'      converts Q to dbl prec 3046
600 CLS:INPUT"Input name/name of parties involved";A$'    check
610 INPUT"Input day each month that payment is due";B$
620 PRINT"Beginning principal =";:PRINTUSINGC$;P
622 PRINT"Interest rate in %  =";:PRINTUSINGC$;I*100
624 PRINT"Monthly payment     =";:PRINTUSINGC$;MP
625 IFX2=0THEN628'                         if MP not yet figured
626 IFX2<>0PRINT"Term of contract    =";:PRINTINT(X2/12)"years,"
X2-INT(X2/12)*12"months
628 PRINT:INPUT"Is all this correct, Y or N";X$:IFX$="N"RETURN
630 LPRINTA$" Land Contract":LPRINTTA"less"D"down ="P"initial pr
incipal":RETURN'                begin LPRINT if input O.K. 3006
640 GOSUB530:LPRINT"Payment of"MP"due each month on "B$:GOSUB530
:LPRINT"       Monthly Principal        New  Interest":LPRINT"Mon
 Interest   Reduced Principal   To date":LPRINT"---   -------
-------  -------   -------":RETURN'                3008
650 LPRINTUSINGM$;X;:LPRINTUSINGC$;MI;MR;P;QT:RETURN'      3060
660 LPRINT"Final payment of";:LPRINTUSINGC$;P+P*I/12;:LPRINT" ne
xt month.":LPRINT"Total interest on land contract will =";:LPRIN
TUSINGC$;QT+P*I/12:GOSUB530:RETURN'                        3070
900 '     *     *     *     *     *         input for subprograms
910 INPUT"Input total amount";TA
920 INPUT"Input down payment";D:P=TA-D:IP=P'  initial principal
930 INPUT"Input % interest";I:I=I/100'              decimal form
940 PRINT"Input monthly payment"
950 INPUT"(If monthly payment is to be figured, input 0)";MP:IFM
P=0THEN5000
1000 '     *     *     *     *     *     *     *     *        menu
1010 PRINT:PRINT"Input 1 to calculate term in months"
1020 PRINT"Input 2 to video print a payment schedule"
1025 PRINT"Input 3 to video print and line print schedule"
1030 PRINT"Input 4 to figure monthly payment, given term"
1040 INPUTX1:ONX1GOTO2000,3000,2090,5000
2000 '     *     *     *     *     *      subprogram 1
2010 X=X+1'                                      number of month
2020 MI=P*I/12'                                 amt for interest
2030 IFMP<MIPRINT"Down payment and/or monthly payment must be in
creased":GOTO900'                  return for different input
2040 MR=MP-MI'                             amt reducing principal
2050 P=P-MR'                                        new principal
2060 IFP<=0PRINTX;"months total to fulfill contract":GOSUB510:GO
TO1000
2070 GOTO2010'                                         next month
2090 FL=1'                                  FLag for LPRINT option
3000 '     *     *     *     *     *     *     *      subprogram 2
3006 IFFL=1GOSUB600:IFX$="N"GOTO900'            for correct input
3008 GOSUB520:PRINT"Payment of"MP"due each month on ";B$:GOSUB52
0:PRINT"       Monthly Principal        New  Interest":PRINT"Mon
Interest   Reduced Principal   To date":PRINT"---    -------   --

-----   -------   -------":IFFL=1GOSUB640
3010 X=X+1
3012 IFF1=2THEN3020'                   P < 6 sig figs, no check req
3014 IFF1=1THEN3018'                     P is dbl prec, is sng req?
3016 IFP>=10000THENF1=1:GOSUB540'     check if P over 6 sig figs
3018 IFP<=10000THENF1=2:GOSUB550'    check if P under 6 sig figs
3020 P2=P*I/12:P2=INT(P2*100+.5)/100:MI=P2'      rounds # to .01
3030 MR=MP-MI
3040 P=P-MR'                                  same as subprogram 1
3044 IFF2=1THEN3050
3046 IFQT>=10000THENF2=1:GOSUB560'  check if QT over 6 sig figs
3050 QT=QT+MI'                             QT=total interest to date
3060 PRINTUSINGM$;X;:PRINTUSINGC$;MI;MR;P;QT:IFFL=1GOSUB650
3070 IFP<=MPPRINT"Final payment of";:PRINTUSINGC$;P+P*I/12;:PRIN
```

Listing 2 continues

carefully name the variables so that their functions are suggested. QT replaces MT to allow defining it as double precision without affecting the other variables beginning with M. Lines 620–628 allow you to verify that all input is correct before printing the payment schedule. I added line 1025 to allow the print option without first running through Subprogram 2, because that can take a long time when term is, for example, 25 years. Flags F1 and F2 avoid unnecessary branching to subroutines.

A Few More Points

When checking your program, note if you require accuracy greater than six significant figures. If so, you will need double-precision values, but double-precision calculations consume so much memory and take so long that you should use them only when required. Subprogram 2 has an instruction to examine the two variables (P and QT) that may involve more than six significant figures and switch them from single to double precision as needed. As the program runs you can easily tell by the speed of the output whether the calculation involves double precision or not.

"Double-precision calculations consume so much memory and take so long that you should use them only when required."

Accuracy in calculations involving money requires that the result be rounded to $0.01 following multiplication or division in order to remove trailing digits before addition or subtraction. Two calculations need this procedure in LANDCONT/BAS—in lines 3020 and 5030. The results of the calculations are immediately rounded to maintain the current degree of precision and trailing digits are removed.

During debugging procedures, check the accuracy of results. There is an easy way to do this: Temporarily define all variables to double precision (DEFDBLA–Z), and then redefine any variable as integral or single precision if you must. (For example, For . . . Next loops cannot work if the variable following For is a double-precision variable.) Use a double-precision type declaration (#) on all constants. Now run your program. It will take a long

Introducing the Most Powerful Business Software Ever!

TRS-80™ (Model I, II, III, or 16) • APPLE™ • IBM™ • OSBORNE™ • CP/M™ • XEROX™

The VERSABUSINESS™ Series

Each VERSABUSINESS module can be purchased and used independently, or can be linked in any combination to form a complete, coordinated business system.

VERSARECEIVABLES™ $99.95

VERSARECEIVABLES™ is a complete menu-driven accounts receivable, invoicing, and monthly statement-generating system. It keeps track of all information related to who owes you or your company money, and can provide automatic billing for past due accounts. VERSARECEIVABLES™ prints all necessary statements, invoices, and summary reports and can be linked with VERSALEDGER II™ and VERSAINVENTORY™.

VERSAPAYABLES™ $99.95

VERSAPAYABLES™ is designed to keep track of current and aged payables, keeping you in touch with all information regarding how much money your company owes, and to whom. VERSAPAYABLES™ maintains a complete record on each vendor, prints checks, check registers, vouchers, transaction reports, aged payables reports, vendor reports, and more. With VERSAPAYABLES™, you can even let your computer automatically select which vouchers are to be paid.

VERSAPAYROLL™ $99.95

VERSAPAYROLL™ is a powerful and sophisticated, but easy to use payroll system that keeps track of all government-required payroll information. Complete employee records are maintained, and all necessary payroll calculations are performed automatically, with totals displayed on screen for operator approval. A payroll can be run totally, automatically, or the operator can intervene to prevent a check from being printed, or to alter information on it. If desired, totals may be posted to the VERSALEDGER II™ system.

VERSAINVENTORY™ $99.95

VERSAINVENTORY™ is a complete inventory control system that gives you instant access to data on any item. VERSAINVENTORY™ keeps track of all information related to what items are in stock, out of stock, on backorder, etc., stores sales and pricing data, alerts you when an item falls below a preset reorder point, and allows you to enter and print invoices directly or to link with the VERSARECEIVABLES™ system. VERSAINVENTORY™ prints all needed inventory listings, reports of items below reorder point, inventory value reports, period and year-to-date sales reports, price lists, inventory checklists, etc.

VERSALEDGER II™ $149.95

VERSALEDGER II™ is a complete accounting system that grows as your business grows. VERSALEDGER II™ can be used as a simple personal checkbook register, expanded to a small business bookkeeping system or developed into a large corporate general ledger system without any additional software.

- VERSALEDGER II™ gives you almost unlimited storage capacity (300 to 10,000 entries per month, depending on the system),
- stores all check and general ledger information forever,
- prints tractor-feed checks,
- handles multiple checkbooks and general ledgers,
- prints 17 customized accounting reports including check registers, balance sheets, income statements, transaction reports, account listings, etc.

VERSALEDGER II™ comes with a professionally-written 160 page manual designed for first-time users. The VERSALEDGER II™ manual will help you become quickly familiar with VERSALEDGER II™, using complete sample data files supplied on diskette and more than 50 pages of sample printouts.

SATISFACTION GUARANTEED!

Every VERSABUSINESS™ module is guaranteed to outperform all other competitive systems, and at a fraction of their cost. If you are not satisfied with any VERSABUSINESS™ module, you may return it within 30 days for a refund. Manuals for any VERSABUSINESS™ module may be purchased for $25 each, credited toward a later purchase of that module.

To Order:
Write or call Toll-free (800) 431-2818
(N.Y.S. residents call 914-425-1535)

- add $3 for shipping in UPS areas
- add $4 for C.O.D. or non-UPS areas
- add $5 to CANADA or MEXICO
- add proper postage elsewhere

DEALER INQUIRIES WELCOME

All prices and specifications subject to change / Delivery subject to availability.

COMPUTRONICS INC.

50 N. PASCACK ROAD, SPRING VALLEY, N.Y. 10977 ~9

* TRS-80 is a trademark of the Radio Shack Division of Tandy Corp. • *APPLE is a trademark of Apple Corp. • *IBM is a trademark of IBM Corp. • *OSBORNE is a trademark of Osborne Corp. *CP/M is a trademark of Digital Research • *XEROX is a trademark of Xerox Corp.

Listing 2 continued

```
T" next month.":PRINT"Total interest on land contract will =";:P
RINTUSINGC$;QT+P*I/12:GOSUB520:IFFL=1GOSUB660:ENDELSE4000
3080 GOTO3010'                                       next month
4000 '    *    *    *    *    *    *    *  LPRINT option
4010 INPUT"Lineprint this, Y or N";X$:IFX$="Y"THENFL=1:F1=0:F2=0
:GOSUB510:GOTO3006'                          FLag for LPRINT option
4020 END
5000 '    *    *    *    *    *    *    *    subprogram 3

5010 INPUT"Input term in years";N:X2=N*12'  term in yrs and mos
5020 I=I*100'                               interest in decimal form
5030 Y=P*I/1200:Z=(1+I/1200)[(12*N):MP=Y/(1-1/Z)' MP by formula
5040 PRINT"Monthly payments =";:PRINTUSINGC$;MP
5050 MP=INT(MP*100+.5)/100'                          round to .01
5060 I=I/100:GOSUB510:GOTO1000'decimal form: restore: goto menu
```

time to execute, but you may compare the results to those previously derived to verify accuracy.

The results of calculations will have the same degree of precision as the most precise operand. Precision is limited by the type declaration of the variable that stores the result. Constants that contain an arithmetic expression (e.g. 1/3) are single precision—accurate to seven significant digits stored, and six displayed or printed. If a single-precision constant is stored in a double-precision variable, the digits following the seventh will be meaningless. You can prevent this by using the double-precision type declaration sign # (1/3# rather than 1/3).

Converting to double precision from single must be done with care because only the first seven digits will be accurate, fewer if the single-precision number contained fewer.

Double-precision values are stored with 17 significant digits, and displayed or printed with 16. (For further information see the reference manual for Level II Basic.)

With the help of Fig. 5, the list of variables, and the remark statements that are added to most of the lines in the completed program, you should understand the purpose of each line of the program.

Debug your program carefully following each modification or addition.

A$	string variable, names of parties
B$	string variable, monthly payment due date
C$	specifies field for PRINTUSING statement
D	down payment
FL	flag: GOTO LPRINT subroutine
F1	flag: is P six significant figures or not
F2	flag: is QT six significant figures or not
I	interest rate: I/100 is decimal form, I*100 returns it to %
IP	initial principal
MI	amount of monthly payment that pays interest
MP	monthly payment
MR	amount of monthly payment that reduces principal
MT	total MI to date (fundamental program only)
M$	specifies field for PRINTUSING statement
N	term in years
P	principal
P2	temporary MI, allows either double or single precision
QT	total MI to date (allows double precision)
TA	total amount
T!	temporary variable, single precision
T#	temporary variable, double precision
X	number of current month
X1	variable integer
X2	term in months
Y	intermediate for MP formula
Z	intermediate for MP formula

formula for subprogram 3:

$$M = \frac{PI/1200}{1 - 1/(1 + I/1200)^{12N}}$$

Fig. 5. Variable List

Contrast Figs. 4 and 5. Particularly note the difference that using double precision makes as well as the appearance of the output.

The five steps are complete; your program is written and debugged. After everything has been satisfactorily accomplished, place a printout of your completed program, including the remark statements, in your reference library. Also, include your other work: the statement of purpose, the organizational chart, the pseudocoded outline, the fundamental program, the list of variables, and any helpful notes you have made. When a question concerning the program arises in the future, or a modification is desirable, it will be easy to recall necessary details by referring to these records.

You may elect to pack the program by eliminating remark statements and by further use of multiple-statement lines. You may use commercial software utilities, or if you have a disk drive and Scripsit, try this excellent method. Save your program in ASCII format (SAVE "LANDCONT/BAS",A) and then load it into Scripsit for easy editing (move blocks of program text, change line numbers, eliminate line numbers by use of multiple-statement lines, and use the global search and replace function to change, for example, Print to LPRINT or the name of a variable throughout the program). When editing is complete save the text in ASCII format again (S,A LANDCONT/BAS) to allow access to the file via Disk Basic. Begin each new line with a line number, and do not leave format instructions in the text or the file will not load. You will get a "Direct Statement in File" error message instead.

Basic

Basic is an acronym for Beginner's All-purpose Symbolic Instruction Code, but do not be deceived by the term "Beginner's" in its name. Basic is a powerful and sophisticated computer language that is appropriately used for a truly amazing variety of purposes. Use it to write individual programs to satisfy your specific needs. Program design and writing is rewarding and enjoyable. With good design procedures you will find it is true in programming as well as in traveling, "Getting there is half the fun!" ■

Spencer Weersing practices dentistry. He can be reached at the Professional Building, Montague, MI 49473.

H&E COMPUTRONICS INC.

• • EVERYTHING FOR YOUR TRS-80™ • ATARI™ • APPLE™ • PET™ • CP/M™ • XEROX™ • IBM™ • OSBORNE™ • •

* TRS-80 is a trademark of the Radio Shack Division of Tandy Corp. - * ATARI is a trademark of Atari Inc. - *APPLE is a trademark of Apple Corp. - * PET is a trademark of Commodore
* CP/M is a trademark of Digital Research - *XEROX is a trademark of Xerox Corp. - * IBM is a trademark of IBM Corp. - * OSBORNE is a trademark of Osborne Corp.

NEW

★ All orders processed within 24-Hours
★ 30-Day money back guarantee

BUSINESS PAC 100

100 Ready-To-Run Business Programs

(ON CASSETTE OR DISKETTE).....Includes 128 Page Users Manual.....
Inventory Control.....Payroll.....Bookkeeping System.....Stock Calculations.....
Checkbook Maintenance.....Accounts Receivable.....Accounts Payable.....

BUSINESS 100 PROGRAM LIST

	NAME	DESCRIPTION
1	RULE78	Interest Apportionment by Rule of the 78's
2	ANNUI	Annuity computation program
3	DATE	Time between dates
4	DAYYEAR	Day of year a particular date falls on
5	LEASEINT	Interest rate on lease
6	BREAKEVN	Breakeven analysis
7	DEPRSL	Straightline depreciation
8	DEPRSY	Sum of the digits depreciation
9	DEPRDB	Declining balance depreciation
10	DEPRDDB	Double declining balance depreciation
11	TAXDEP	Cash flow vs. depreciation tables
12	CHECK2	Prints NEBS checks along with daily register
13	CHECKBK1	Checkbook maintenance program
14	MORTGAGE/A	Mortgage amortization table
15	MULTMON	Computes time needed for money to double, triple, etc.
16	SALVAGE	Determines salvage value of an investment
17	RRVARIN	Rate of return on investment with variable inflows
18	RRCONST	Rate of return on investment with constant inflows
19	EFFECT	Effective interest rate of a loan
20	FVAL	Future value of an investment (compound interest)
21	PVAL	Present value of a future amount
22	LOANPAY	Amount of payment on a loan
23	REGWITH	Equal withdrawals from investment to leave 0 over
24	SIMPDISK	Simple discount analysis
25	DATEVAL	Equivalent & nonequivalent dated values for oblig.
26	ANNUDEF	Present value of deferred annuities
27	MARKUP	% Markup analysis for items
28	SINKFUND	Sinking fund amortization program
29	BONDVAL	Value of a bond
30	DEPLETE	Depletion analysis
31	BLACKSH	Black Scholes options analysis
32	STOCVAL1	Expected return on stock via discounts dividends
33	WARVAL	Value of a warrant
34	BONDVAL2	Value of a bond
35	EPSEST	Estimate of future earnings per share for company
36	BETAALPH	Computes alpha and beta variables for stock
37	SHARPE1	Portfolio selection model-i.e. what stocks to hold
38	OPTWRITE	Option writing computations
39	RTVAL	Value of a right
40	EXPVAL	Expected value analysis
41	BAYES	Bayesian decisions
42	VALPRINF	Value of perfect information
43	VALADINF	Value of additional information
44	UTILITY	Derives utility function
45	SIMPLEX	Linear programming solution by simplex method
46	TRANS	Transportation method for linear programming
47	EOQ	Economic order quantity inventory model
48	QUEUE1	Single server queueing (waiting line) model
49	CVP	Cost-volume-profit analysis
50	CONDPROF	Conditional profit tables
51	OPTLOSS	Opportunity loss tables
52	FQUOQ	Fixed quantity economic order quantity model
53	FQEOWSH	As above but with shortages permitted
54	FQEOQPB	As above but with quantity price breaks
55	QUEUECB	Cost-benefit waiting line analysis
56	NCFANAL	Net cash-flow analysis for simple investment
57	PROFIND	Profitability index of a project
58	CAP1	Cap. Asset Pr. Model analysis of project
59	WACC	Weighted average cost of capital
60	COMPBAL	True rate on loan with compensating bal. required
61	DISCBAL	True rate on discounted loan
62	MERGANAL	Merger analysis computations
63	FINRAT	Financial ratios for a firm
64	NPV	Net present value of project
65	PRINDLAS	Laspeyres price index
66	PRINDPA	Paasche price index
67	SEASIND	Constructs seasonal quantity indices for company
68	TIMETR	Time series analysis linear trend
69	TIMEMOV	Time series analysis moving average trend
70	FUPRINF	Future price estimation with inflation
71	MAILPAC	Mailing list system
72	LETWRT	Letter writing system-links with MAILPAC
73	SORT3	Sorts list of names
74	LABEL1	Shipping label maker
75	LABEL2	Name label maker
76	BUSBUD	DOME business bookkeeping system
77	TIMECLCK	Computes weeks total hours from timeclock info.
78	ACCTPAY	In memory accounts payable system-storage permitted
79	INVOICE	Generate invoice on screen and print on printer
80	INVENT2	In memory inventory control system
81	TELDIR	Computerized telephone directory
82	TIMUSAN	Time use analysis
83	ASSIGN	Use of assignment algorithm for optimal job assign.
84	ACCTREC	In memory accounts receivable system-storage ok
85	TERMSPAY	Compares 3 methods of repayment of loans
86	PAYNET	Computes gross pay required for given net
87	SELLPR	Computes selling price for given after tax amount
88	ARBCOMP	Arbitrage computations
89	DEPRSF	Sinking fund depreciation
90	UPSZONE	Finds UPS zones from zip code
91	ENVELOPE	Types envelope including return address
92	AUTOEXP	Automobile expense analysis
93	INSFILE	Insurance policy file
94	PAYROLL2	In memory payroll system
95	DILANAL	Dilution analysis
96	LOANAFFD	Loan amount a borrower can afford
97	RENTPRCH	Purchase price for rental property
98	SALELEAS	Sale-leaseback analysis
99	RRCONVBD	Investor's rate of return on convertable bond
100	PORTVAL9	Stock market portfolio storage-valuation program

☐ TRS-80 Cassette Version	$99.95
☐ TRS-80 (Mod-I or III), Pet, Apple or Atari Versions	$99.95
☐ TRS-80 Mod-II, IBM, Osborne and CP/M Versions	$149.95

ADD $3.00 FOR SHIPPING IN UPS AREAS
ADD $4.00 FOR C.O.D. OR NON-UPS AREAS
ADD $5.00 TO CANADA AND MEXICO
ADD PROPER POSTAGE OUTSIDE OF U.S., CANADA AND MEXICO

NEW TOLL-FREE ORDER LINE
(OUTSIDE OF N.Y. STATE)
(800) 431-2818

AMERICAN EXPRESS

VISA

H&E COMPUTRONICS INC.
MATHEMATICAL APPLICATIONS SERVICE™

50 N. PASCACK ROAD
SPRING VALLEY, NEW YORK 10977

master charge

24 HOUR ORDER LINE
(914) 425-1535

ASK FOR OUR 64-PAGE CATALOG

DEALER INQUIRIES INVITED

ALL PRICES & SPECIFICATIONS SUBJECT TO CHANGE
DELIVERY SUBJECT TO AVAILABILITY

See List of Advertisers on Page 563

Linear Programming

by David Clapp

This method is usually used by mainframes to solve complex problems. Dr. Clapp gives a simple demonstration, so you can use it, too.

Linear programming is a powerful technique for analyzing decision problems. The technique has been in use for many years, although primarily for analysis of large-scale problems on large computers. While large computers are necessary for analyzing some linear programming problems, you can solve many complex problems using a simple algorithm on a microcomputer.

I will describe a simple linear programming routine that you can use to solve maximization and minimization problems. The size of the problem that may be solved is limited only by your microcomputer's memory. To enlarge the program's capacity, change only the dimension statement.

Linear Programming

Linear programming is a relatively complex mathematical procedure. I will explain the technique by reference to simple examples.

Consider the ABC Toy Company. This company (as with most) is interested in making a profit from its operations. Assume that ABC manufactures only two toys: xylophones and yoyos. For convenience, label the number of each produced in a production period as X and Y, respectively. To make the maximum profit, ABC should make an infinite number of each toy, sell them, and accumulate an infinite amount of cash.

In reality, however, no company can make an unlimited number of anything. The availability of raw materials, production capacity of machines, storage space for produced products as well as investment capital all are examples of restrictions on total possible production. These restrictions are called constraints in a linear programming problem. The goal of linear programming is to allocate limited resources to production in order to maximize profit.

For the ABC problem, I have built some simple, arbitrary constraints. For example, assume that only two machines are used in production (machine A and machine B). The number of production hours on each machine is limited by hours in the day as well as by skilled operator time.

For our example, assume that machine A is available only for 1,000 hours per production period (per week, for example) and machine B is available for 800 hours per period. It takes twice as long to produce a xylophone than a yoyo on machine A, while it requires equal time for each toy on machine B. These production restrictions can be expressed mathematically as:

$$2X + Y \leq 1000$$
$$X + Y \leq 800$$

The expressions above are called inequalities since production can be any value up to and equal to the hours available. Making these expressions inequalities allows a wide variation of possible solutions. If they are equalities (e.g., $2X + Y = 1000$), all available hours on machine A must be used. Clearly, the inequalities make more sense since it may be more economical not to use all of the machine capacity available.

As additional simple constraints on this problem, we have specified that no more than 400 xylophones or 700 yoyos can be made in any production period. This type of constraint could be the result of expected sales, available raw materials, or available investment capital. These constraints may also be expressed mathematically as $X \leq 400$ and $Y \leq 700$.

Again, the inequalities make sense since we may decide to produce up to the limit, but still at some lower level.

Finally, we must express the objective of our decision problem—to make the most money. This expression, called the objective function, is expressed mathematically as:

$$\text{Maximize } Z = 0.4X + 0.3Y$$

This expression reflects the selling price of each toy (40 cents for xylophones and 30 cents for yoyos), and our purpose (to maximize). Thus, Z can be any value depending on how many toys we make. If we make ten of each toy then Z equals $7; hence, the objective function expresses the value of each possible solution in dollars. We wish this to be the greatest amount possible, subject to the constraints.

The decision problem described so far can be summarized as:

$$\text{Maximize } Z = 0.4X + 0.3Y$$
subject to:
$$\begin{aligned} 2X + Y &\leq 1000 \\ X + Y &\leq 800 \\ X &\leq 400 \\ Y &\leq 700 \end{aligned}$$

The Key Box

Model I
16K RAM
Basic Level II

The only solutions we are interested in have positive variable values (we cannot manufacture a negative number) and integer or whole number solutions (we cannot manufacture fractions of yoyos or xylophones).

Regarding the integer constraints, it so happens that all the solutions of interest in this problem are whole numbers, or integers. This is not always the case. If non-integer solutions did exist, an integer programming algorithm would be required. In many linear programming problems, non-integer solutions are perfectly acceptable (see the nutrition example I will give later).

The easiest way to visualize the ABC Toy solution is with a graph. Figure 1 shows a plot of the constraints for this problem. Note that the constraints are simply straight lines on the graph relating numbers of X we can make to numbers of Y. All solutions must be within the dotted lines (called the feasible region) to satisfy the problem's constraints.

The objective function is not a single line, but a "family" of lines depending on the value of Z. Two typical values of the objective function are plotted (for Z equals \$120 and Z equals \$240) as solid lines on the graph. These are called "iso-profit" lines, since the profit (Z) is the same at any point along the line. For the Z-equals-\$120 line, the profit is \$120 at either end (X equals 0 and Y equals 400 or X equals 300 and Y equals 0) and at any point between.

You can make several important observations by looking at the objective function lines. First, the value of Z increases as the lines move away from the origin (the 0,0 point). Also, the best solutions occur only at corners of the feasible region.

This is the key to the linear programming algorithm—the best solutions occur at corners. We want to move the objective function line as far from the origin as possible (getting larger values of Z, or profit) and still just barely touch the feasible region. Clearly, this can only happen at corners. Imagine a ruler parallel to the sketched objective function lines. This ruler is moved out as far as possible, remaining parallel to the plotted lines, to achieve the highest Z. The best solution must be at a corner and not at any interior point.

Looking at the sketch, there are six possible solutions or corners to choose from. These are in Table 1. The solution (4) produces a profit of \$260, yet remains within the constraints of the problem. You can see this solution by moving the imaginary ruler from the origin. Point 4 is the furthest the ruler can travel parallel to the sketched objective function lines and still touch the feasible region.

The fact that the solutions to the problem must occur at corners suggests the mathematical method to solve this problem. Corners in the graph are intersections of lines, or simultaneous solutions to linear equations. For example, point 4 is the solution to the

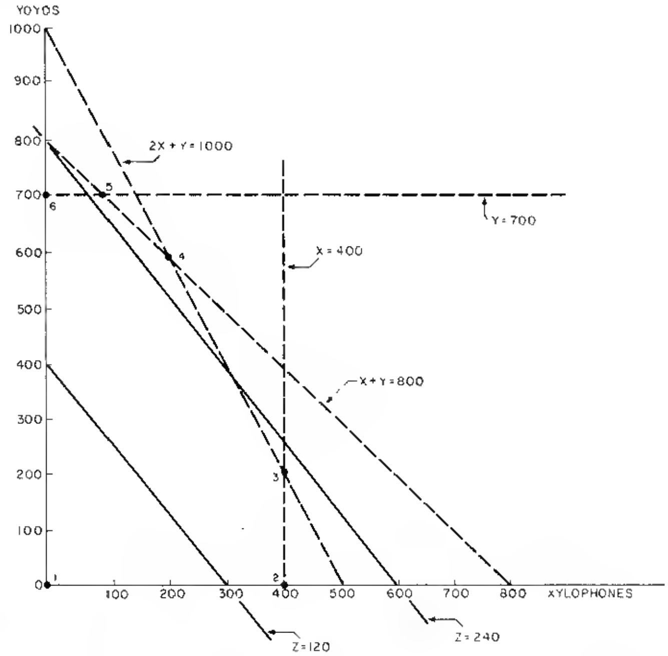


Figure 1

See List of Advertisers on Page 563

equations:

$$\begin{aligned} 2X + Y &= 1000 \\ X + Y &= 800 \end{aligned}$$

You can solve these equations algebraically and find the solution X equals 200 and Y equals 600, which we confirmed graphically earlier (for which Z equals $260).

The purpose of linear programming, then, is to find solutions of systems of linear equations or find solutions at each corner of the feasible region. The linear programming algorithm achieves this logically by moving from one corner to the next until it finds the highest value of Z.

The first step in setting up a linear programming problem is defining the objective function and constraints for the decision problem under study. In our ABC problem, we have already completed this step. The second step is to convert the inequalities to equations:

$$\begin{aligned} 2X + Y + M &= 1000 \\ X + Y + N &= 800 \\ X + O &= 400 \\ Y + P &= 700 \end{aligned}$$

Note the slack variables M, N, O, and P, aptly named since they take up the slack in converting the inequalities to equalities. (We must do this because there is no algebraic way to solve inequalities.) In the final solution the slack variables express the amount of unused resource. For our optimal solution of X equals 200 and Y equals 600 there is no slack in the first two constraints (or M and N equal zero); however, there is slack in the last two constraints since we are not making 400 xylophones or 700 yoyos. In the final solution, O equals 200 and P equals 100. The meaning of slack variables should become clearer as we move on.

The equations are difficult to solve algebraically. We can only solve systems of equations where the number of

1. Make no toys	profit = $0
2. Make 400 xylophones	profit = $160
3. Make 200 yoyos, 400 xylophones	profit = $220
4. Make 600 yoyos, 200 xylophones	profit = $260
5. Make 700 yoyos, 100 xylophones	profit = $250
6. Make 700 yoyos	profit = $210

Table 1

```
10 CLEAR 500
20 DEFSNG A-Z
30 DIM D(20,20),P(20),IV(20),SC(20)
40 CLS
50 PRINT TAB(23) "LINEAR PROGRAMMING"
60 PRINT
70 REM ENTER DATA BY APPENDING DATA STATEMENTS TO THIS PROGRAM
80 REM 1ST DATA STMT CONTAINS NUMBER OF ROW, NUMBER OF COLUMNS,AND NUMBER O
F REAL (ACTUAL DECISION) VARIABLES IN THE PROBLEM
90 REM 2ND DATA STMT CONTAINS COEFFICIENTS OF OBJECTIVE FUNTION
100 REM REMAINING DATA STMTS CONTAIN COEFFICIENTS OF EACH ROW IN THE INITIA
L TABLEAU INCLUDING RHS.  A SIMPLE EXAMPLE IS INCLUDED WITH THIS LISTING
110 READ IW,IZ,IY
120 IX=IZ-1
130 INPUT"DO YOU WISH SUCCESSIVE TABLEAUX PRINTED (YES/NO)";IB$
140 FOR M=1 TO IX
150 READ P(M)
160 NEXT M
170 FOR M=1 TO IW
180 FOR N= 1 TO IZ
190 READ D(M,N)
200 NEXT N
210 NEXT M
220 INPUT"DO YOU WANT OUTPUT TO GO TO THE PRINTER (YES/NO)";PO$
230 IF LEFT$(PO$,1)="Y" OR LEFT$(PO$,1)="y" THEN PO=1
240 IF PO=1 THEN PRINT"PRINTING...":A1=PEEK(16414):A2=PEEK(16415):POKE 1641
4,PEEK(16422):POKE 16415,PEEK(16423)
250 PRINT" "
260 PRINT "THE INITIAL TABLEAU IS:"
270 FOR N=1 TO IX
280 PRINT USING "#########";N;
290 NEXT N
300 PRINT
310 FOR M=1 TO IX
320 PRINT USING "#####.##";P(M);
330 NEXT M
340 PRINT
350 FOR M=1 TO IW
360 FOR N=1 TO IZ
370 PRINT USING "#####.##";D(M,N);
380 NEXT N
390 PRINT
400 NEXT M
410 FOR N=IY TO IX
420 FOR L=1 TO IW
430 IF D(L,N) = 1 THEN 460
440 NEXT L
450 GOTO 470
460 IV(L)=N
470 NEXT N
480 Z=0
490 FOR M=1 TO IW
500 IM=IV(M)
510 Z=Z+D(M,IZ)*P(IM)
520 NEXT M
530 NP=1
540 SX=0
550 FOR N=1 TO IX
560 FOR I=1 TO IW
570 IF N=IV(I) THEN 680
580 NEXT I
590 SM=0
600 FOR I=1 TO IW
610 J=IV(I)
620 SM=SM+P(J)*D(I,N)
630 NEXT I
640 SC(N)=SM-P(N)
650 IF SC(N)>=SX THEN 680
660 SX=SC(N)
670 IC=N
680 NEXT N
690 FOR M=1 TO IW
700 IM=IV(M)
710 SC(IM)=0
720 NEXT M
730 IF LEFT$(IB$,1)<>"Y" AND LEFT$(IB$,1)<>"y" THEN 920
740 PRINT" "
750 PRINT "TABLEAU NO.":NP
760 FOR N=1 TO IX
770 PRINT USING "#########":N;
780 NEXT N
790 PRINT
800 FOR M=1 TO IW
810 FOR N=1 TO IZ
820 PRINT USING"#####.##";D(M,N);
830 NEXT N
840 PRINT
850 NEXT M
860 FOR J= 1 TO IX
870 PRINT USING"#####.##";SC(J);
880 NEXT J
890 PRINT
900 PRINT "THE OBJECTIVE FUNCTION VALUE IS";Z
910 NP=NP+1
920 IF SX>=0 THEN 1200
930 SV=999999
940 IR=-1
950 FOR M=1 TO IW
960 IF D(M,IC)<=0 THEN 1010
970 QU=D(M,IZ)/D(M,IC)
980 IF (QU-SV)>=0 THEN 1010
990 IR=M
1000 SV=QU
1010 NEXT M
1020 IF IR=-1 THEN PRINT "UNBOUNDED SOLUTION":GOTO 1270
1030 IV(IR)=IC
1040 DI=D(IR,IC)
1050 FOR N=1 TO IZ
1060 CK=D(IR,N)
1070 D(IR,N)=CK/DI
1080 NEXT N
1090 FOR M=1 TO IW
1100 IF M=IR THEN 1170
1110 CM=-D(M,IC)
1120 FOR N=1 TO IZ
1130 TM=D(IR,N)*CM
1140 SK=D(M,N)
1150 D(M,N)=SK+TM
1160 NEXT N
1170 NEXT M
1180 Z=Z-SV*SX
1190 GOTO 540
1200 PRINT"
1210 PRINT "THE OPTIMAL TABLEAU IS:"
1220 FOR M=1 TO IW
1230 A$=RIGHT$(STR$(IV(M)),LEN(STR$(IV(M)))-1)
1240 PRINT "X("A$") = "USING"#####.##";D(M,IZ)
1250 NEXT M
1260 PRINT "THE OBJECTIVE FUNCTION VALUE IS";Z
1270 IF PO=1 THEN POKE 16414, A1 : POKE 16415, A2 : CLS
1280 END
1290 DATA 3,7,3
1300 DATA -2,-1,0,-100,-100,0
1310 DATA 3,1,0,1,0,0,3
1320 DATA 4,3,-1,0,1,0,6
1330 DATA 1,2,0,0,0,1,3
```

Program Listing

equations equals the number of unknowns. Above there are six unknowns and only four equations. The theory of linear programming suggests considering only m variables for solutions at any given time, where m is the number of equations. In our example, we want to have only four variables under consideration at a time so that we can calculate a unique solution with the four available equations. This set of variables is called the "basis."

An initial basis to start with would be to set the variables X and Y equal to zero and solve for the remaining four variables. Looking at the equations above, this particular solution is M equals 1000, N equals 800, O equals 400 and P equals 700. This solution is the "do nothing" alternative; we are making no toys and profit is $0. Furthermore, all production is at slack levels: None of the resources are used at all. This solution corresponds to point 1 in the graph.

Obviously, we wish to make some toys. What variable should now come into the solution and which should go out (only four can be in the basis at any time)? We make more money from xylophones, so common sense suggests trying to make some X. Pivoting, the process of bringing the variable X into the solution and taking out another variable, is especially amenable to computer solution.

The Program

Following the simple instructions in the comment lines in the Program Listing, set up the data for this problem as:

```
1400 DATA 4,7,3
1410 DATA .4,.3,0,0,0,0
1420 DATA 2,1,1,0,0,0,1000
1430 DATA 1,1,0,1,0,0,800
1440 DATA 1,0,0,0,1,0,400
1450 DATA 0,1,0,0,0,1,700
```

Merge this data with the program with the Basic Merge command or by typing the data statements into the program directly. You could easily modify the program to accept data via an Input command, but with a disk system and Merge you can rapidly input different sets of fairly complex data to the program.

The program will respond with the output shown in Fig. 2. The initial tableau simply places the input data in convenient columnar format. Check this data carefully with the statement of the problem. The objective function is displayed on the top row of the tableau. You can identify the current solution by locating the identity matrix. The identity matrix is a set of columns of zeros and ones which looks like:

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

In the initial tableau, the identity matrix is the rightmost four columns under variable numbers 3, 4, 5 and 6, or the variables M, N, O and P. This corresponds to the solution in which we produce no toys.

The second tableau is identical to the first except the objective function has been arrayed on the bottom row with negative signs. This prepares the analysis for selecting the first pivot point. This row is not always identical to the objective function row in the initial tableau; in minimization problems (as in the nutrition problem below), this row is quite different.

The selection of the pivot point requires only two checks. First, check for the most negative coefficient in the bottom row to identify the most promising incoming variable (xylophones, in our example). Next, check for the outgoing variable by forming a ratio of the right side over the coefficient in the pivot column, and choose the element

The Initial Tableau Is:

1	2	3	4	5	6	
0.40	0.30	0.00	0.00	0.00	0.00	
2.00	1.00	1.00	0.00	0.00	0.00	1000.00
1.00	1.00	0.00	1.00	0.00	0.00	800.00
1.00	0.00	0.00	0.00	1.00	0.00	400.00
0.00	1.00	0.00	0.00	0.00	1.00	700.00

Tableau 1

1	2	3	4	5	6	
2.00	1.00	1.00	0.00	0.00	0.00	1000.00
1.00	1.00	0.00	1.00	0.00	0.00	800.00
1.00	0.00	0.00	0.00	1.00	0.00	400.00
0.00	1.00	0.00	0.00	0.00	1.00	700.00
−0.40	−0.30	0.00	0.00	0.00	0.00	

The objective function value is 0

Tableau 2

1	2	3	4	5	6	
0.00	**1.00**	1.00	0.00	−2.00	0.00	200.00
0.00	1.00	0.00	1.00	−1.00	0.00	400.00
1.00	0.00	0.00	0.00	1.00	0.00	400.00
0.00	1.00	0.00	0.00	0.00	1.00	700.00
0.00	−0.30	0.00	0.00	0.40	0.00	

The objective function value is 160

Tableau 3

1	2	3	4	5	6	
0.00	1.00	1.00	0.00	−2.00	0.00	200.00
0.00	0.00	−1.00	1.00	**1.00**	0.00	200.00
1.00	0.00	0.00	0.00	1.00	0.00	400.00
0.00	0.00	−1.00	0.00	2.00	1.00	500.00
0.00	0.00	0.30	0.00	−0.20	0.00	

The objective function value is 220

Tableau 4

1	2	3	4	5	6	
0.00	1.00	−1.00	2.00	0.00	0.00	600.00
0.00	0.00	−1.00	1.00	1.00	0.00	200.00
1.00	0.00	1.00	−1.00	0.00	0.00	200.00
0.00	0.00	1.00	−2.00	0.00	1.00	100.00
0.00	0.00	0.10	0.20	0.00	0.00	

The objective function value is 260

The Optimal Tableau Is:
X(2) = 600.00
X(5) = 200.00
X(1) = 200.00
X(6) = 100.00
The objective function value is 260

Figure 2

that produces the lowest ratio.

Examine Fig. 2 to identify the pivot point in Tableau 1. Look for the pivot in column 1 (xylophones) since -0.4 is the smallest coefficient. Next, form the ratios 1000/2, 800/1, and 400/1. The smallest value is 400. The pivot point in tableau 1 is in bold type. When finding ratios, always ignore negative ratios or division by zero.

You are now ready to develop tableau 2. Your objective is to produce a column of zeros and a one under column 1 (that is, create a column of the identity matrix). The one must appear at the pivot point. By examining column 1 in tableau 2, you can see the desired column of zeros and a one. To obtain this column use the pivot row to eliminate variable one from all other rows in the tableau. Thus, to eliminate the one from row 2 in tableau 1, subtract the entire row 3 (pivot row) from row 2. (This is equivalent to algebra's Gauss-Jordan Reduction, where we subtract one equation from another to solve systems of equations.)

Subtraction continues until zeros are produced in the pivot column, including the objective function row. Eliminate the two in row 1 by multiplying row 3 by two and subtracting the result from row 1. In all cases, the pivot element must be a one before beginning these operations. If it is not, simply divide through the equation to produce a one before beginning the row operations. To fully understand the solution, subtract all rows in tableau 1 to produce tableau 2.

The linear programming algorithm now continues until there are no more negative coefficients in the bottom row. For the ABC Toy problem this occurs at tableau 4, which displays the solution 600, 200, 200, and 100. To identify which variables in tableau 4 correspond to which solution, look for the identity matrix (it will not be in order). Also, the program reprints the solution with the appropriate variable identified. This solution corresponds to the graphical solution shown earlier.

It is interesting to follow the route of the algorithm on Fig. 1 as it moves from corner to corner of the feasible region. As expected, tableau 1 is the origin (point 1 on the graph). Then, the solution moves to points 2, 3 and 4 (tableaus 2, 3 and 4). Calculations terminate at point 4 because a move to point 5 only results in a lower profit. The theory of linear programming includes a mathematical proof that point 4 is the optimal point and any further search will be fruitless. Such a characteristic is a real boon since the solution must occur after a series of finite steps. The technique is a powerful analytical application for computer processing.

Nutrition

To further explain linear programming and interpretation of the output, I will give another example. This nutrition problem provides an opportunity to present additional features, including minimization and the use of artificial variables. This problem is a lot closer to reality than the ABC Toy Co.

Nutritionists have determined that an adult must consume at least 75 grams of protein, 90 grams of fats and 300 grams of carbohydrates daily to maintain good health. Table 2 lists several common foods with protein, fat and carbohydrate content, as well as an approximate cost. All data is grams per ounce of the food item. Suppose we want to produce a meal which will supply these substances at the lowest cost. This is a minimization problem; the formulation should look like Fig. 3.

$$\text{Minimize: } Z = 0.5X_1 + .7X_2 + 3.75X_3 + 1X_4 + 1.5X_5 + 1.5X_6 + 1.75_7 + 1.125_8$$

subject to:

$$2.4X_1 + 7.1X_2 + 8.4X_3 + 6.2X_4 + .7X_5 + 2.3X_6 + 1.6X_7 \geqslant$$
$$.3X_1 + 9.8X_2 + 2.1X_3 + .2X_4 + \quad + 9.4X_6 + 3.7X_7 + 28.1X_8 \geqslant 90$$
$$15.8X_1 \quad + 11.4X_5 + .9X_6 + 18.0X_7 \geqslant 300$$

Figure 3

Item	Protein	Fat	Carbohydrate	Cost
Wheat bread	2.4	0.3	15.8	0.5
Cheddar cheese	7.1	9.8	0	1.7
Roast chicken	8.4	2.1	0	3.75
Steamed haddock	6.2	0.2	0	1.0
Dried prunes	0.7	0	11.4	1.5
Walnuts	2.3	9.4	0.9	1.5
Gingerbread	1.6	3.7	18.0	1.75
Margarine	0	28.1	0	1.125

Table 2

Special techniques are needed to deal with a minimization problem as defined in Fig. 3. First, the computer program is designed to maximize; you must trick it into minimizing by multiplying through the objective function with a -1. The program will try to make the result as negatively large as possible (or as small as possible).

We have other problems with the constraints. Consider Constraint 3, for example:

$$15.8X_1 + 11.4X_5 + .9X_6 + 18X_7 - A = 300$$

Here, we converted the inequality to an equality by subtracting a slack variable. This variable is now called a surplus variable since it is the amount of carbohydrate in excess of the minimum requirements. All seems well, except the sign of the surplus is negative and the algorithm must have positive signs for the starting variables in the identity matrix. We overcome this by adding an artificial variable to the inequality. This variable is included only to provide a suitable starting variable. Since it is useless other than to get us started, we wish to discard it as soon as possible.

We discard artificial variables rapidly by giving them extremely large coefficients in the objective function. The program rapidly excludes them from the solution and they will never return since they cost so much relative to other variables (remember that this is a minimization problem and we want to keep costs as low as possible).

The results of our handiwork are shown in Fig. 4. Here, the input data is displayed along with the tableaus. Notice first in the objective function data (line 1410) that the surplus variables have a coefficient of zero as expected, but the artificial variables have a coefficient of 100 (extremely expensive items). All coefficients are negative since this is a minimization problem. Also note in the initial tableau that the initial basis consists of the artificial variables (X_{10}, X_{12} and X_{14}) in the columns headed by the -100 values.

The objective function row is considerably changed in tableau 1. Here, as in the ABC Toy problem, we wish to find the most negative coefficient in

DISCOUNT

A LITTLE BIT OF "GREEN" GOE$ A LONG WAY!!!

FREE SHIPPING *on all prepaid cash orders over $50.

COMPARE OUR PRICES...Call (216) 481-1600

DISKIT III

MODEL III DISK UPGRADE

Features
- Uses Micro Main Frame Finest Disk Controller
- Gold Plated Edge Connectors
- Switching Power Supply
- Supports 5" or 8" Drives
- 40/80 Track Supported
- Single/Dual Head Supported
- Metal Disk Drive Brackets
- All Hardware and Cables for Two Disk Drives
- 1 Hour or Less for Installation
- 100% Compatible
- No Soldering Needed
- 180 Days Warranty on Controller

DISKIT III W/D Drives	$279.00
DISKIT III W/ONE Tandon 100-1 40 Track Drive	$479.00
DISKIT III W/TWO Tandon 100-1 40 Track Drive	$699.00

DEALER INQUIRIES WELCOME

DISKIT III is a trademark of MDS

PRINTERS
NEW LOWER PRICES

Epson MX-80	$CALL
Epson MX-80FT w/Graftrax	$CALL
Epson MX-100 w/Graftrax	$CALL
IDS Prism 80	$CALL
IDS Prism 132	$CALL
Okidata Microline 80	$CALL
Okidata Microline 82A	$CALL
Okidata Microline 83A	$CALL
Okidata Microline 84	$CALL
Prowriter 8510	$CALL
Smith Corona TP-1	$629.00

TRS-80, HEATH, IBM, APPLE, ZENITH DISK DRIVES
AT OR BELOW DEALER'S COST!

TRAXX-Model I & III Complete	
Tandon 40 Track	$279.00
Tandon 80 Track	$CALL
★ ★ ★ ★ ★ SPECIAL ★ ★ ★ ★ ★	
Tandon 80/80 Track	$429.00
BARE TANDON	
T100-1 40 Track	$209.00
T100-2 40/40 Track	$CALL
T100-3 80 Track	$CALL
★ ★ ★ ★ ★ SPECIAL ★ ★ ★ ★ ★	
T100-4 80/80 Track	$359.00
TRAXX 8 INCH MODELS	
Seaman 77 Track s/s	$339.00
NEW TANDON "THIN LINE" 8 INCH	
848-1 Single Side	$399.00
848-2 Dual Side	$499.00
APPLE Add-On Disk Drive	$359.00

MODEMS

UDS 103 LP Direct Connect	$175.00
UDS 103 JLP AUTO ANS	$209.00
Hayes Micromodem II (APPLE)	$299.00
Hayes 100 Model (S-100)	$325.00
Hayes Smart Modem (RS-232)	$249.00

MICROBUFFER

MBP-16K Parallel	$149.00
MBS-8K Serial	$149.00
INTERFACE CARDS	
8141 (RS-232)	$75.00
8150 (2K Buffer RS-232)	$150.00
8161 (IEEE 488)	$55.00
8131 (Apple Card)	$85.00
8230 (Apple Cable)	$25.00
8220 (TRS-80 Cable)	$25.00

DISKETTES

PARAGON MAGNETICS GOLD, Soft-Sectored, Single-Sided, Double-Density 5¼ inch diskettes with reinforcing HUB-RINGS	$23.95
VERBATIM-Soft-Sectored Diskettes	
5¼" 1S/DDen (MD525-01)	$26.95
5¼" 2S/DDen (MD550-01)	$39.95
5¼" 2S/4Den (MD557-01)	$51.50
8" 1S/DDen (FD34-8000)	$43.95
VERBATIM-Hard-Sectored Diskettes	
5¼" 1S/DDen 10-Sector (MD525-10)	$26.95
5¼" 2S/DDen 10-Sector (MD550-10)	$39.95
5¼" 2S/4Den 10-Sector (MD557-10)	$51.50

SUPPLIES

HUB RING KIT for 5¼" disks	$10.95
HUB RING KIT for 8" disks	$12.95
REFILLS (50 Hub Rings)	$ 5.95
CLEANING KIT for 5¼" drives	$24.95
5¼" Diskette Case	$ 3.50
8" Diskette Case	$ 3.95
5¼" File Box for 50 diskettes	$24.95
AVERY TABULABLES	
1,000 3½" x 15/16	$ 5.49
3,000 3½ x 15/16	10.95
5,000 3½ x 15/16	$15.95
FAN FOLD PAPER (Prices F.O.B. S.P.)	
9½ x 11 18 lb. WHITE 3,000 ct.	$27.95
14⅞ x 11 18 lb WHITE 3,000 ct.	$37.95

BOOKS

OTHER MYSTERIES	
TRS-80 DISK	$19.95
Microsoft Basic Decoded	$24.95
The Custom TRS-80	$28.95
Basic Faster & Better	$29.95
The Custom Apple	$28.95
1001 Things To Do With Your Personal Computer	$10.95
Choosing A Word Processor	$15.95
CP/M Primer	$15.95
I Speak Basic TRS-80	$12.95

TRS-80

Model III w/2 Tandon 48K	$1795.00
Model III w/1 Tandon 48K	$1599.00

Both units come with 120 day warranty from MDS and use our MEMORY and DISKIT

LNW

LNW-80 Computer	$1595.00
LNW-80-II Computer w/cpm 96K	$1995.00
SYSTEM EXPANSION II	$ 349.00
LNDOUBLER 5/8 Board	$ 149.00

SOFTWARE

TRS-80 Model I & III

★ ★ ★ ★ ★ SPECIAL ★ ★ ★ ★ ★

FREE Floppy Doctor with the purchase of any DISKIT III .... $30.00 VALUE!

NEWDOS 80 2.0	$139.95
DOSPLUS 3.4 Model I	$ 89.00
DOSPLUS 3.3 Model I	$ 69.95
NEW Electric Pencil	$79.95
ELECTRIC WEBSTER	
50,000 Word Spelling Checker	$ 79.95
Correcting Feature	$ 59.95
When purchased together	$129.95
UNITERM Terminal	$ 79.95
UNITERM/80 Terminal	$ 89.95
ACE MAIL for Hayes Smart Modem	$ 69.95
LABELMAKER for MX80	$ 19.50
AIDS III Data Management	$ 49.95
Maxi Manager	$ 89.00
Floppy Doctor I, III	$ 24.95
Inventory +	$ 29.95
Cash Register/80	$ 29.95

GAMES

TRS-80 • SAVE • MOST GAMES AVAILABLE FOR APPLE

No.	Title	Media	Price
141-001	Introductory 3-Pack Rescue, Morloc's and Dalestones	DISK	$49.95
221-001	Super Nova	DISK	$17.95
221-002	Galaxy Invasion	DISK	$17.95
221-003	Attack Force	DISK	$17.95
221-004	Cosmic Fighter	DISK	$17.95
221-006	Robot Attack	DISK	$17.95
221-008	Stellar Escort	DISK	$17.95
351-001	Scarfman (Mod. I/III)	DISK	$17.95
351-500	Scarfman (color)	CASS	$17.95
351-501	New Bounceoids	DISK	$19.95
391-001	Math-Pak-1	CASS	$14.95
391-002	Math-Pak-2	CASS	$14.95
431-001	Sargon II (Mod. I)	DISK	$29.95
431-101	Sargon II (Mod. III)	DISK	$34.95
511-003	Adventure	DISK	$29.95
021-004	Pigskin	DISK	$17.95
060-601	Adventure #1,2&3	DISK	$34.95
060-604	Adventure #4,5&6	DISK	$34.95
060-607	Adventure #7,8&9	DISK	$34.95
060-611	Adventure #10,11&12	DISK	$34.95
061-066	The Mean Checker Machine	DISK	$17.95

MDS MICRO DATA SUPPLIES

22295 EUCLID AVE.
EUCLID, OHIO 44117
(216) 481-1600

459

WE ACCEPT
- Visa
- MasterCard
- Checks
- Money Order
- C.O.D.

ALL PRICES ARE FOR MAIL ORDER ONLY. Prices, Specifications and Offerings subject to change without notice

ADD $3.00 FOR SHIPPING & HANDLING. $6.00 Extra for C.O.D. Orders. Ohio Residents add 6.5% Sales Tax

the objective function row which will reveal the most promising incoming variable. In this case, variable X_8 (margarine) is the incoming variable. In column 8 there is only one choice for the pivot (90/28.1) and X_{12} is hence the outgoing variable. (This is one of the artificial variables we wish to banish.)

Finally, in Fig. 4, tableau 8 (the last tableau) and the final solution are displayed. (Six other tableaus computed on the way to this solution were omitted to save space.) Observing the objective function row in tableau 8, no further negative signs are seen and the optimal solution is reached. The solution is 18.99 grams of bread, 4.75 grams of fish, and 2.97 grams of margarine. Looks like a fish sandwich! You can question the proportions of items, but the solution does satisfy the constraints.

In the real world, an analyst would go back to the original problem and devise additional constraints. (For example, we may never want more than 1/2 gram of margarine in any solution.)

The objective function row in the last tableau, now called "shadow prices," reveals the necessary cost reduction for a food item to come into the solution. A person particularly fond of chicken sandwiches (versus fish) could examine the shadow price for chicken ($2.32). If the price were reduced to $1.42, chicken would come into the solution. (It would then have a negative coefficient in bottom row.) You may wish to change the cost of chicken in the initial data and explore the resulting solution.

You may wish to explore more alternatives regarding the final solution to the Toy Co. or nutrition problems. Additional solutions are possible. The computer program permits rapid reanalysis of the problem, eliminating the drudgery of hand solution.

Here are some additional problems for interested readers. Explore the changes in the ABC Toy problem if all available machine hours *must* be used. (Hint: This requirement implies equality constraints and you must add an artificial variable to the formulation to get started.) What happens when you make the following changes to the nutrition problem?

Modify the input data and resolve the problem by reducing the price of chicken by the amount of the shadow price ($2.33).

Supposing your diet must contain exactly 215 grams of carbohydrates, find the new solution.

Suppose the protein requirement is revised to only 50 grams or more of protein. Can you guess from the current solution if the diet will change? In either case, resolve the problem. ■

Dr. David Clapp is an engineer with a federal agency. He can be reached at 1769 Kingsway Court, Cincinnati, OH 45230.

The Initial Tableau Is:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
−0.50	−1.70	−3.75	−1.00	−1.50	−1.50	−1.75	−1.13	0.00	−100.00	0.00	−100.00	0.00	−100.00	
2.40	7.10	8.40	6.20	0.70	2.30	1.60	0.00	−1.00	1.00	0.00	0.00	0.00	0.00	75.00
0.30	9.80	2.10	0.20	0.00	9.40	3.70	28.10	0.00	0.00	−1.00	1.00	0.00	0.00	90.00
15.80	0.00	0.00	0.00	11.40	0.90	18.00	0.00	0.00	0.00	0.00	0.00	−1.00	1.00	300.00

Tableau 1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
2.40	7.10	8.40	6.20	0.70	2.30	1.60	0.00	−1.00	1.00	0.00	0.00	0.00	0.00	75.00
0.30	9.80	2.10	0.20	0.00	9.40	3.70	28.10	0.00	0.00	−1.00	1.00	0.00	0.00	90.00
15.80	0.00	0.00	0.00	11.40	0.90	18.00	0.00	0.00	0.00	0.00	0.00	−1.00	1.00	300.00
−1849.50	−1688.30	−1046.25	−639.00	−1208.50	−1258.50	−2328.25	−2808.88	100.00	0.00	100.00	0.00	100.00	0.00	

The objective function value is −46500

Tableau 2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
2.40	7.10	8.40	6.20	0.70	2.30	1.60	0.00	−1.00	1.00	0.00	0.00	0.00	0.00	75.00
0.01	0.35	0.07	0.01	0.00	0.33	0.13	1.00	0.00	0.00	−0.04	0.04	0.00	0.00	3.20
15.80	0.00	0.00	0.00	11.40	0.90	18.00	0.00	0.00	0.00	0.00	0.00	−1.00	1.00	300.00
−1819.51	−708.69	−836.33	−619.01	−1208.50	−318.88	−1958.40	0.00	100.00	0.00	0.04	99.96	100.00	0.00	

The objective function value is −37503.6

Tableau 7

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
1.00	0.00	0.00	0.00	0.72	0.06	1.14	0.00	0.00	0.00	0.00	0.00	−0.06	0.06	18.99
0.00	1.00	1.18	0.87	−0.15	0.30	−0.16	0.00	−0.14	0.14	0.00	−0.00	0.02	−0.02	4.15
0.00	0.00	−0.34	−0.30	0.04	0.23	0.18	1.00	0.05	−0.05	−0.04	0.04	−0.01	0.01	1.55
0.00	0.00	2.12	−0.15	1.34	0.70	1.25	0.00	0.18	99.82	0.04	99.96	0.00	100.00	

The objective function value is −18.2915

Tableau 8

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
1.00	−0.00	−0.00	0.00	0.72	0.06	1.14	0.00	0.00	−0.00	−0.00	0.00	−0.06	0.06	18.99
0.00	1.15	1.35	1.00	−0.17	0.35	−0.18	0.00	−0.16	0.16	0.00	−0.00	0.02	−0.02	4.75
0.00	0.34	0.07	0.00	−0.01	0.33	0.12	1.00	0.00	−0.00	−0.04	0.04	0.00	−0.00	2.97
0.00	0.17	2.32	0.00	1.31	0.75	1.23	0.00	0.16	99.84	0.04	99.96	0.01	99.99	

The objective function value is −17.58

The Optimal Tableau Is:
X(1) = 18.99
X(4) = 4.75
X(8) = 2.97
The objective function value is −17.58

Figure 4

— Professional —

REAL ESTATE SOFTWARE

[illegible] CPM SYSTEMS

• PROPERTY MANAGEMENT SYSTEM [illegible]

[illegible]

• PROPERTY LISTINGS/COMPARABLES [illegible]

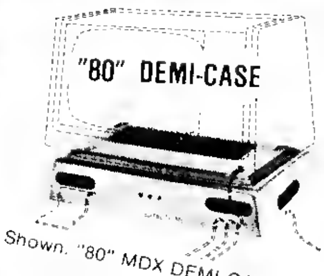
[illegible]

• REAL ESTATE ANALYSIS MODULES [illegible]

[illegible]

• WORD PROCESSING — [illegible]

[illegible]



174

Encloses LNW & MDX I & II
P.C. Boards, plus
Cabling, A/C Cords, etc.
Silver-gray Fiberglass

$32.50 plus $2.50 shipping
N.Y.S. residents add 7% tax

SYRACUSE R&D CENTER
Box 125, Dewitt, N.Y. 13214
"Specializing in Electronic Packaging"

OPTIONS—80, NOW FOR APPLE* & TRS—80*

OPTION INVESTING

PROGRAM ANALYSES RETURN FROM LISTED STOCK OPTION INVESTING

HANDLES CALLS, PUTS, SPREADS, IN AND OUT; COMMISSIONS, RISK, COST OF MONEY, DIVIDENDS. TABLES AND GRAPHS. PRINTS, STORES TO DISK. INDEXED MANUAL A COMPLETE GUIDE TO OPTION INVESTING. M/C & VISA. $125. SEND FOR FREE BROCHURE.
OPTIONS—80, BOX 471
CONCORD, MASS 01742
*TRADEMARK TANDY CORP #TRADEMARK APPLE COMPUTER, INC

YOUR TRS-80®...OUR BASIC BOOK, THE PERFECT COUPLE

LEARNING TRS-80 BASIC starts where the Radio Shack® getting started manuals end, fulfilling the needs of elementary and intermediate level users. If you own or have access to a Model I, II/16, or III, our 544 page BASIC book is a necessity.

IT'S SIMPLE

We receive testimonials every day commenting on the simplicity of this book.

- "very well organized... careful step-by-step progression" **—R.G., Oxnard, CA**
- "extremely logical progression and excellent explanations" **—J.S., Pacifica, CA**
- "easy reading style"**—J.B., St. Louis Park, MN**

In fact we think it is so simple that....

WE GUARANTEE IT

Try **LEARNING TRS-80 BASIC** free for 15 days and if you don't agree that it is as simple as we and users like yourself are saying, then send back the book in saleable condition and pay nothing.

Call and order today. **(800) 854-6505** our 24 hour order hot line
(714) 588-0996 our Calif. (8am-5pm) order line

Please send me **LEARNING TRS-80 BASIC** on a 15 day **FREE** examination. At the end of that time, I will send payment, plus postage and handling, or return the book and owe nothing. On all prepaid orders, Visa or MasterCard charges, publisher pays postage and handling, send $19.95 only (Calif. residents add 6%) —same return guarantee. Offer good in USA only. Payment must accompany orders from P.O. Box numbers. Name of individual ordering must be filled in.

NAME ______________________

ADDRESS ______________________

CITY/STATE/ZIP ______________________

☐ CHECK ENCLOSED ☐ VISA ☐ MASTERCARD ACCOUNT # ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ EXPIRATION DATE ☐☐☐☐

SIGNATURE ______________________

COMPUSOFT® PUBLISHING
P.O. Box 19669 Dept. F San Diego, CA 92119

CompuSoft Books are available at your local computer and book stores. Allow 4 to 6 weeks for delivery.

TRS-80 and Radio Shack are registered Trademarks of Tandy Corporation

365

Model III Relative Files

by Andrew Rucks

The drawbacks of direct access files can be overcome by using relative files accessed by the scatter technique with open addressing.

It's difficult to write application programs using direct-access files for the Model III. Model III Disk Basic offers constructs for creating and accessing random or direct-access files; however, it doesn't permit efficient use of file space unless logical records conform to the size parameter of physical records.

Using the R option with a file-open statement designates a direct-access file. For the Model III and most other microcomputers, direct access means each of the physical records allocated to a file can be accessed independently. For example, the fifth record of a file containing 10 records can be accessed without having to access the first record, second record and so forth. You can determine the number of physical records allocated to a file by examining the #Rec column of the report produced by a DIR command.

Model III Disk Organization and Buffering

Disk Basic running under TRSDOS stores data on disks in fixed-length blocks. These fixed-length blocks, called sectors, are the units of addressable space on a disk. Therefore sector and physical record are synonyms. The size (length) of a sector is expressed as the number of characters (bytes) that can be stored in a sector. On Model III disks the sector size is 256 characters. The combination of sector size, number of sectors per track, and the number of tracks on a disk determine a disk's storage capacity.

For example, the 5¼ inch disks designed for use with Model III systems have 40 tracks, 18 sectors per track, and 256 bytes per sector, and therefore can store 184,326 characters. Actual data storage capacity of a disk is less than the physical capacity of a disk because track 17 of a disk is reserved for the disk's directory and cannot be used for data storage.

The Model III and other microcomputers reserve a portion of random-access memory (RAM) for disk input-output (I/O) operations. These reserved areas are called buffers. A buffer has the same capacity (size) as a sector or physical record. When a disk input operation is performed, a copy of the information stored in a sector is placed in a buffer where it becomes available to an application program. A disk output operation is the reverse of this process. For every disk input operation a complete sector is read into memory and for every disk output operation a complete sector is written onto the disk.

The full sector read/write applies even when you answer V to the "How many files?" query upon entering Disk Basic. The use of variable-length records has no perceivable impact on the manner in which TRSDOS references files. Disk Basic through TRSDOS reads an entire sector; however, when this data is transferred to a buffer, only the first N bytes are placed in the buffer—the remainder are disregarded.

For example, assume that a program is using a variable-length record file and the file's buffer is fielded for 128 characters. When a disk read operation is performed on the file, a full sector (256 characters) is read from the disk but only the leftmost 128 characters are placed in the file buffer. The remaining 128 characters of the sector are ignored. This remainder cannot be used for data storage because Disk Basic and TRSDOS cannot reference fragments of sectors.

Impact of Fixed Blocks

Fixed-block disk organization is not limited to the Model III; it is also common in other microcomputers, minicomputers and large-scale systems. Operating systems of computers in the mini and large-scale classes perform buffer fielding and data transfer without programmer intervention. These operating systems can write logical records across sector boundaries, thus optimizing disk space utilization. Microcomputers cannot place more than one logical record in a sector without programmer intervention.

Consider writing eight records of 32 characters each in a direct-access file on the Model III compared with the same operation on a typical minicomputer. Storing the eight records on the Model III requires 2,048 characters of disk space (eight records of 256 characters each). For the minicomputer only 256 characters of disk space are required. The Model III consumes eight times more disk space than the minicom-

The Key Box

Model III
Disk System

```
100   DIM BUS$(8)
        "
        "
        "
1000 OPEN "R",1,"FILE"
1010 FIELD 1, 32 AS BU$(1), 32 AS BU$(2),
     32 AS BU$(3), 32 AS BU$(4), 32 AS
     BU$(5), 32 AS BU$(6), 32 AS BU$(7),
     32 AS BU$(8)
```

Figure 1a

```
100   DIM BU$(5)
        "
        "
        "
1000 OPEN "R",1,"FILE"
1010 FIELD 1, 48 AS BU$(1), 48 AS BU$(2),
     48 AS BU$(3), 48 AS BU$(4), 48 AS
     BU$(5), 16 AS LO$
```

Figure 1b

puter to store the same amount of information.

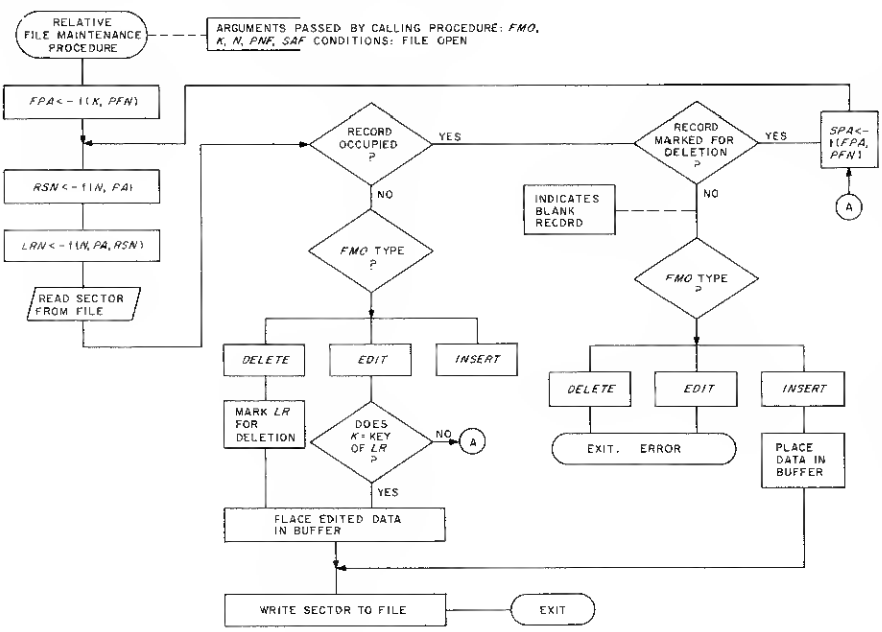
Model III programmers have several options available to solve the problem of wasted disk space associated with direct-access files. One solution is to avoid the use of direct-access files by relying only on sequential files. Using sequential files improves disk space usage efficiency; however, this technique significantly increases the search time for records and complicates file maintenance procedures. Record search-time inefficiency results from having to read n–1 records in order to reach the nth record in a file. In a sequential file maintenance procedure, the original (master file) must be merged with a transaction file (containing changes, deletions or insertions for the master file) to create an update of the master file. The transaction file must be sorted in the same order as the master file.

Relative Files

In relative files the location of a logical record in a file is determined by a value within the record referred to as the record key (e.g., employee ID in a payroll system). The location of a logical record in the file is relative to the contents of the record. Relative file organization treats a file as a pool of space wherein records can be stored at random or scattered throughout the file.

To create a storage pool, field the buffer areas of direct-access files as elements of an array. The size of the array is the integer portion of the quotient obtained by dividing the number of bytes in each logical record into the buffer size. The treatment of buffers as arrays creates a pool of storage area in much the same way that soft-drink cartons subdivide the space in a soft-drink case.

Figures 1a and 1b show two examples of fielding a file buffer as an array. In Fig. 1a, the logical record length is 32 characters. Therefore, eight logical records can be placed in the buffer and in each sector allocated to the file. Figure 1b illustrates the fielding of a buffer in situations where sector size is not evenly divisible by the logical record length. In Fig. 1b, the logical record length is 48 characters, resulting in five logical records being held in the buffer and a slack or waste area of 16 characters.



Legend:

FMO = file maintenance operation (delete, edit, insert)
FPA = first probe address
K = record key
LRN = logical record number
N = number of logical records per sector
PA = probe address (either FPA or SPA)
PNF = prime number factor—largest prime number less than the number of logical records per sector times the number of sectors allocated to a file divided by 0.95
RSN = relative sector number (1< =RSN< =SAF)
SAF = sectors allocated to a file multiplied by 0.95

Fig. 2. File Maintenance Procedure—Scatter Method with Open Addressing

Determining Logical Record Location

The actual location of a record in a relative file is determined by a formula known as a hash function. Hash functions produce a set of sector numbers that fall within an allowable range. This range is expressed as $1 <= Sf <= Sn$, where 1 is the minimum relative sector number, Sn is the number of sectors allocated to a file and is the largest relative sector number, and Sf is the sector number produced by the hash function.

A good hash function minimizes the probability that the hashing of two independent keys will produce the same relative sector number. The generation of duplicate addresses is termed collision. Since hashing algorithms that totally avoid collisions are rare, you must use a method for resolving these situations.

Open addressing is one of several methods for resolving collisions and offers advantages to the Model III user not provided by alternative methods. Open addressing minimizes the number of disk accesses required to determine if a record is present in a file. Although open addressing involves more computational overhead than other collision-resolution techniques, this overhead is insignificant when compared to the overhead involved in accessing a Model III disk.

In open addressing a record key is hashed repetitively and the resulting sector number is probed for the appropriate record. This set of sector numbers is referred to as the probe sequence. Members of a probe sequence are generated on an as-needed basis: When the probe of a logical record does not achieve the desired result, the key is hashed again to produce the next member of the probe sequence. An important feature of open addressing is that the formula used to generate a probe sequence always generates the same set of sector numbers for a given record key.

```
200 REM *** First Probe Address Function
210 DEF FNFP (K,P)=K-(INT(K/P)*P)+1
220 REM   where, K=record key and P=prime number factor

230 REM *** Subsequent Probe Address Function
240 DEF FNSP (PA,K,P)=(PA+(K-(INT(K/P)*P))-
        (INT (PA+(K-INT(K/P)*P)/P)*P)
250 REM   where, K and P are defined above and PA=probe address

260 REM *** Relative Sector Number Function
270 DEF FNRS(PA,N)=INT (PA-1/N)+1
280 REM   where, PA is defined above and N=number of logical records per sector

290 REM *** Logical Record Number Function
300 DEF FNLR(SA,N,PA)=SA*N-(PA-1)
310 REM   where, SA=relative sector number, and N and PA are defined above
```

Program Listing

File Maintenance

File maintenance involves three possible actions: placing a new record in a file (insert), changing the contents of a record in a file (edit) and removing a record from a file (delete). Figure 2 shows a flowchart of the procedure for performing file maintenance using the scatter method with open addressing.

Sizing a Relative File

Use a file load factor of approximately 95 percent for this implementation of the relative organization. A 95-percent file load factor means you should plan to use only about 95 percent of the space allocated to a relative access file. The slack prevents the processing time of a search from significantly increasing as the file becomes full. For example, assume you are creating a relative file with 1,000 logical records of 45 bytes each. Five logical records can be stored in each sector (256/45 = 5 and a slack area of 31 characters), and 200 sectors are required for data storage. However, because of the file load factor, you should allocate five percent more sectors to the file, thereby making the file 210 sectors long.

"Slack prevents the processing time of a search from significantly increasing as the file becomes full."

Hash Functions

The scatter technique with open addressing requires two hash functions: one to determine the first probe address (FPA) and another to determine all subsequent probe addresses (SPA). The FPA function is: FPA = modulo (record key,P) + 1. Record key is the value of the record key and P is the largest prime number less than or equal to the number of logical records multiplied by the number of sectors allocated to the file multiplied by the file load factor.

The function for determining SPAs is: SPA = modulo(PA + modulo (record key/P,P),P) + 1. PA is the previous unsuccessful probe address determined by either the FPA or SPA functions. Record key and P are defined in the FPA function. Probe addresses produced by these algorithms have the characteristic of 1 < = probe address (PA) < = P.

Identifying Logical Records

A probe address is a composite of the number of sectors and the number of logical records per sector. It must be reduced to a relative sector number (RSN) and a logical record number (LRN). The RSN identifies the sector that will be read into a file buffer and examined for the desired logical record. The RSN function is: RSN = integer((PA - 1) / N) + 1. PA is the probe address and N is the number of logical records per sector.

The final function required to implement the relative file organization is one to determine the element within the buffer array that contains the logical record. The logical record number (LRN) function is: LRN = (RSN * N) - (PA - 1). RSN, N and PA are the same as defined in previous functions.

Disk Basic Implementation

The program segment presented in the listing is written in Disk Basic. This segment illustrates the define function (DEF FN) construct for each of the functions presented in this article. ■

Peter Ginter and Andrew Rucks can be reached at the College of Business Administration, University of Arkansas, Fayetteville, AK 72701.

Towards Better Programming

by John T. Blair

Improve your programming technique by applying these straightforward methods—Program Development Language and the Scientific Method.

Professional tools and ideas are always welcome to newcomers to computers and programming. I will present a procedure for solving problems (the scientific method) and Program Development Language (PDL, similar to flowcharting) for program design.

These techniques are not restricted to the TRS-80 or to Basic. They can be applied to a program in any language run on any computer. For programming in Assembly language, these principles are almost mandatory.

I will use these techniques to write a Basic program which will convert an Assembly-language source program to an ASCII file, and an ASCII file back to an Assembly-language source program. Many TRS-80 terminal programs only transmit or receive ASCII files. Consequently, Assembly-language source programs cannot be sent via these terminal programs unless converted to ASCII files.

Getting Started

Three distinct steps must be performed to get a program from thought to execution.

The first phase is developing the program specifications. You must define requirements and identify possible problem areas. Next comes the program design, where logic flow is developed via a PDL or flowcharts. This is the most important part of writing a program, because a poorly designed program costs extra time and money. Here is where you find out:

Do I want to do this?; Can I do this? (Do I have the time and resources?); and What assumptions have been made by the person requesting the program?

The second phase is translating ideas from the PDL into the language of the computer.

The last phase consists of running the program, debugging it, and looking for logic flaws.

Each of these independent steps could be handled by a separate person. As hobbyists, most of us must do all three. In an academic environment, most courses emphasize language syntax; the student is expected to learn programming design by default.

Scientific Method

The scientific method is a systematic thought process for solving problems or predicting the correct results of an action. *The Nature of Scientific Thought* states the scientific method in its simplest form as:

> Postulate a model based on existing experimental observation or measurement. Check the predictions of the model against further observation or measurements. Adjust or replace the model as required by the new observation or measurement.

How does this relate to programming? The programmer is trying to predict a computer's action when given a sequence of instructions. The scientific method leads to a logical path (flowchart or PDL), a way to predict if the program will yield the desired results. Expanding and redefining the

Initial problem analysis

- Sift through the material or data, to produce a better definition of the given data.
- Define the requirements of the problem. In other words, what are we required to find?

Attempt a general model

- Draw a general system flowchart. Examine the logic.
- Draw detailed or subordinate flowcharts. All language should be machine independent. Write short, descriptive English words to describe functions.
- Test the final model by implementing the coding in the language to be used.

Adjusting the model

- Look for bugs.
- Retest and look for bugs again until complete.

Table 1. Scientific Programming

The Key Box

Disk Basic
Model I or III
32K RAM
TRSDOS
One Disk Drive

basic steps of this methodology and applying it to programming, the solution of a problem can be divided into several parts (see Table 1).

This is called "top down design," or structured programming. Be aware of the difference between designing and writing a program. Designing a program is the development of a smooth flow of ideas and a step-by-step procedure (algorithm) to solve a problem. It is not necessary for the algorithm to run on a computer. The logic flow can be checked by reading the PDL or following the lines in a flowchart. Once the logic has been proved, the design can be translated into a language your computer will understand. Notice that the program design is machine independent! Writing a program is putting instructions in a language the com-

Program Listing

```
10 '         *****    C O N M L S     ****
15 '
20 '      CONVERT MACHINE LANGUAGE SOURCE TO AN ASCII DATA FILE
30 '     BY  JOHN T. BLAIR  WA4OHZ   DATE 06/14/80
40 '                    LAST REV 11/01/80
50 '    END-OF-FILES      BASIC = 0D(H)     EDTASM = 0D1A(H)
52 '                     PENCIL = 00(H)
60 '    A TAB FOR  EDTASM = 09(H)
70 '
80 '      DEFINATION OF VARIABLES USED
90 '  I$  = INPUT BUFFER
100 '  IN$ = INPUT FILE
110 '  N$ = FILE NAME LESS EXTENSION
120 '  NI$ = DRIVE NUMBER (INPUT)
130 '  NO$ = DRIVE NUMBER (OUTPUT)
140 '  O$  = OUTPUT BUFFER
150 '  OU$ = OUTPUT FILE
160 '  LN  = LINE NUMBER  (OUTPUT FILE)
170 '  S$ = SPACE STRING
172 '  T$ = TEMP OUTPUT LINE
180 '
190 CLEAR 1000:DIM I$(256),O$(256): ' CLEAR STRING SPACE FOR INP
UT & OUTPUT BUFFERS
200 S$=STRING$(10,32)
210 '
220 '       CRT  HEADERS
230 '
240 CLS:PRINT"                          C O N M L S"
250 PRINT:PRINT"THIS PROGRAM WILL CONVERT A MACHINE LANGUAGE SOU
RCE PROGRAM"
260 PRINT"TO AN ASCII FILE OR VISA VERSA. EXTENSIONS ARE N-O-T R
EQUIRED,";
270 PRINT"BUT MUST BE:":PRINT "FOR THE INPUT FILESPEC."
280 PRINT"    TXT  =  SOURCE FILE
290 PRINT"    ASC  =  ASCII FILE."
300 PRINT:PRINT:PRINT"WHICH TYPE OF FILE IS THE INPUT FILE?"
310 PRINT"   1 = M/L SOURCE FILE":PRINT"   2 = ASCII FILE"
320 INPUT"INPUT YOUR OPTION #";O
330 INPUT "       SOURCE FILESPEC:";IN$
340 INPUT "       SOURCE DRIVE #";NI$
350 INPUT "       DEST   DRIVE #";NO$
360 '
370 '     GET FILE NAME FROM FILESPEC
380 '
390 J = INSTR(IN$,"/") : IF J = 0 THEN 400 ELSE 410
400 J=LEN(IN$):GOTO420
410 J=J-1
420 ON O GOTO 500,1220
430 '
440 '*******************************************************
450 '**********     CONVERT SOURCE FILE TO ASCII FILE   ******
460 '*******************************************************
470 '
480 '       DISK INITILAIZATION
490 '
500 OU$=MID$(IN$,1,J)+"/ASC"+":"+NO$
510 IN$=MID$(IN$,1,J)+"/TXT"+":"+NI$
520 O$=""
530 OPEN "I",1,IN$:'               INPUT FILE
540 OPEN "O",2,OU$:'               OUTPUT FILE
550 CLS:PRINT"NEW FILE: ";OU$
560 '
570 '  READ  1ST  LINE FROM DISK INTO INPUT BUFFER
580 '
590 I$="":LINEINPUT #1,I$
600 GOSUB 970:IF L=0 THEN 820
610 L=LEN(I$):I$=MID$(I$,8,L):'  STRIP OFF FILE NAME
620 O$=""
630 GOSUB 880:'                    CONVERT AND ADD LINE NUMBER
640 GOSUB 1050:'                   CONVERT TABS
650 PRINT O$: '                    PRINT TEXT LINE ON CRT
660 PRINT #2, O$: '                OUTPUT TEXT LINE TO DISK
670 '
680 '   READ REMAINING LINES FROM DISK
690 '
700 I$="":O$="":'                  CLEAR INPUT & OUTPUT BUFFERS
710 LINEINPUT #1,I$: '             READ  NTH  TEXT LINE
720 IF EOF(1) THEN 820
730    GOSUB 970:IF L=0 THEN 820
740    GOSUB 880: '                CONVERT AND ADD LINE NUMBER
750    GOSUB 1050:'                CHECK FOR TABS
760    PRINT O$
770    PRINT #2, O$
780    GOTO 700:'                  REPEAT  TILL E-O-F
790 '
800 '     CLOSE FILES & ADD END OF TEXT
802 '  THIS ASC FILE IS DIRECTLY READABLE BY PENCIL
810 '
820 O$="":O$=O$+CHR$(13)+CHR$(00):'       ADD  EOT  CHAR
830 PRINT #2, O$
840 GOTO 1670
850 '
860 '  CONVERT BIT 8 HIGH ASCII TO BIT 8 LOW
870 '
880 FOR I=1 TO 5: '                5 DIGITS / LINE NUMBER
890    O$=O$+CHR$(ASC(MID$(I$,I,1)) AND 127)
900 NEXT I
910 O$=O$+CHR$(32):'               ADD <SP> APT LINE #
920 I$=MID$(I$,7,L):L=LEN(I$):'  MODIFY INPUT STRING
930 RETURN
940 '
950 '           END-OF-TEXT FINDER
952 '
954 '   THIS SECTION LOOKS FOR A  1A(H) THE EDTASM'S  EOT.
960 '
970 J=INSTR(I$,CHR$(26))
980 IF J=0 THEN 1000:'            J = 0 ==>  NO EOF THIS READ
990    I$=MID$(I$,1,J-1)
1000   L = LEN(I$)
1010 RETURN
1020 '
1030 '           TAB CONVERTER
1032 '
1034 ' THIS SECTION SCANS EACH LINE LOOKING FOR  09(H) &
1036 ' CONVERTS THEM TO THE APPROPRIATE NUMBER OF <SP>.
1040 '
1050 S=1:T$="":'                        SET STARTING POS
1060 J=INSTR(MID$(I$,S,L),CHR$(9)):'    LOCATE A TAB
1070 IF J=0 THEN 1140;'                 NO TABS
1080   J1=(INT(J/8)+1)*8:'              CAL NEXT TAB STOP
1090   J2=J1-J:'                        # <SP> REQ.
1100   IF J=1 THEN T$=T$+MID$(S$,1,8):GOTO1120
1110       T$=T$+MID$(I$,S,J-1)+MID$(S$,1,J2+1)
1120       S=S+J:'                      BUMP START POS PTR.
1130   GOTO 1060
1140 O$=O$+T$+MID$(I$,S,L):'             BUILD OUTPUT LINE
1150 RETURN
1160 '*******************************************************
1170 '********  CONVERT ASCII FILE TO SOURCE  *************
1180 '*******************************************************
1190 '
1200 '          DISK INITIALIZATION
1210 '
1220 IN$=MID$(IN$,1,J)+"/ASC"+NI$
1230 OU$=MID$(IN$,1,J)+"/TXT"+":"+NO$
1240 O$="":I$=""
1250 OPEN "I",1,IN$:'             INPUT FILE
1260 OPEN "O",2,OU$:'             OUTPUT FILE
1270 CLS:PRINT"NEW FILE: ";OU$
1280 '
1290 '  READ 1ST SECTOR FROM DISK INTO BUFFER
1300 '
1310 I$="":O$="": '               CLEAR INPUT & OUTPUT BUFFER
1320 LINEINPUT #1,I$
1330 O$=CHR$(211):'               FILE  IDENT  CODE (CARRY OVER F
ROM TAPE)
1340 '
1350 ' INSERT FILE NAME IN OUTPUT BUFFER
1360 '
1370 N=6-J:'                      # SPACES TO FILL OUT FILE NAME
1380 IF N=0 THEN 1410:'           FILE NAME 6 CHAR LONG
1390    O$=O$+MID$(IN$,1,J)+MID$(S$,1,N)
1400    GOTO 1420
1410 O$=O$+MID$(IN$,1,J):'        ADD NAME TO OUTPUT BUFFER
1420 GOSUB 1720:'                 CONVERT AND ADD LINE NUMBER
1430 L=LEN(I$)
1440 I$=MID$(I$,6,L):'            STRIP OFF LINE NUMBER FROM INPU
T BUFFER
1450 O$=O$+I$:'                   ADD REMAINDER OF TEXT LINE
1460 PRINT O$
1470 PRINT #2,O$
1480 '
1490 '           READ REMAINING SECTORS FROM DISK
1500 '
1510 I$="":O$="":'                CLEAR INPUT & OUTPUT BUFFERS
1520 LINEINPUT #1,I$:'            GET  NTH TEXT LINE
1530 IF EOF(1) THEN 1650
1540    GOSUB 1790
1550    IF L=0 THEN 1650
1560    GOSUB 1720:'              CONVERT AND ADD LINE NUMBER
1570    L=LEN(I$)
1580    O$=O$+MID$(I$,6,L):'      ADD REMAINDER OF TEXT LINE
1590    PRINT O$
1600    PRINT #2,O$
1610 GOTO 1510:'                  REPEAT TILL E-O-F
1620 '
1630 '     CLOSE FILES & EXIT
1640 '
1650 O$="":O$=O$+CHR$(26):'       ADD  EOT  CHAR
1660 PRINT #2, O$
1670 CLOSE ALL
1680 END
1690 '
1700 '  CONVERT BIT 0 LOW ASCII TO BIT 8 HIGH
1710 '
1720 FOR I=1 TO 5:
1730    O$=O$+CHR$(ASC(MID$(I$,I,1)) OR 128)
1740 NEXT I
1750 RETURN
1760 '
1770 '          END-OF-TEXT FINDER
1780 '
1790 J=INSTR(I$,CHR$(13))
1800 IF J=0 THEN 1820:'           J = 0 ==>  NO EOF THIS READ
1810    I$=MID$(I$,1,J-1)
1820    L = LEN(I$)
1830 RETURN: '  *** END OF PROG ***
```

puter can understand.

Solving a Programming Problem

First, read and understand the problem. The hardest part of this is understanding the problem.

Second, restate the problem in your own words, and draw pictures if possible.

Third, identify all given information. Identify any areas where further research is required. Eliminate all extraneous information, and state all assumptions. State all external files or algorithms not explicitly given in the problem.

In many cases the problem will be too large to solve. To reduce the problem size, certain pieces of information may be ignored or taken for granted. These are thc assumptions. If when the program is executed the solution yields incorrect results, you can check your assumptions. The particular solution may be applicable to similar problems. While many problems are similar, the results differ depending on the assumptions.

Fourth, identify all the information you are required to find. Identify error handling/trapping ("goofproofing"), find the various intermediate results, and finally, determine what was originally required of the problem.

Many problems have several items which may or may not be order dependent: These are the intermediate results. The final result is the answer to the problem. Error trapping is commonly overlooked. What happens if the user inputs the wrong data? The program should require the operator to reenter data.

Fifth, design the program. Use flowcharts or a PDL in plain English (not machine or language dependent) and describe the function to be performed. Check the logic of the program design.

Sixth, code the design into a particular program language. At this point the program becomes machine dependent unless a higher-level language is used.

Seventh, analyze the results for accuracy. Execute the program to verify results. When running the program in this debugging stage, use sample data. If the results are wrong, check the design again. If the design is good, check the coding.

The most common error is in a loop and is called "off by one." This is when a routine to be repeated X times is executed one more or less. Again run some sample data. If it is to be repeated 100 times, try setting the counter to five. If the loop is only executed four times, you must increment the counter by one. Conversely if it is executed six times, you must decrement the counter by one.

The best method for determining where an error occurs is to use the "divide and conquer" technique. In machine language use break points, and in a high-level language use print statements. Choose some arbitrary point and insert a print statement or break point, and output the results to this point. If the answers are correct, move further down the program. If they are wrong, move towards the beginning of the program.

Rules for Structured Programming

First, as you design a program start with the overall picture, then write the smaller sections. By dividing a large program into several small modules (subroutines) the task does not seem as overpowering. By keeping it simple, you will not get hung up trying to do too much at once.

"A Programming Development Language (PDL) is a verbal picture of the program function, much like a flowchart."

Second, use subroutines as small independent modules.

Third, each module should perform only one function. If you want a module to perform a function that has several subfunctions, the subfunctions should be called from the first module. If each module performs only one function, adding or deleting functions to the program is simple. This avoids having two routines do the same thing.

Fourth, each module should have *only* one entry and exit point. This prevents a module from performing more than one function. It also helps the programmer keep track of any flags that have been set.

Fifth, avoid self-modifying coding. Self-modifying code writes over itself to change instructions. It is more common in Assembly or machine language, and cannot always be avoided.

Sixth, variables should have meaningful names. In a high-level language (Basic) or Assembly language, use a variable name resembling what is stored.

Seventh, initialize and define all variables at the beginning of the program. Many newer languages require that all variables be allocated at the beginning of the program. In Basic it doesn't matter. Each variable should be given an initial value at the beginning of the program, so the computer can allocate storage for it. A remark statement should tell what the variable is used for. Suppose you have an array "grades." It appears that the contents will be a student's grades. But where are the grades for quizzes, mid-term and final exams, and projects located in the array? How will you or anyone else know where to modify the program?

Eighth, assign a variable name to constants. In a high-level program or low-level language this allows the program more flexibility. If the data must be changed, you just alter the value once at the beginning of the program, instead of at each place it is used.

Programming Development Language

A Programming Development Language (PDL) is a verbal picture of the program function, much like a flowchart. A PDL is a set of instructions used to design a program. It is both language and machine independent. This means it is not Fortran, Basic, Cobol, nor does it run on a Z80, 6800, 6502, PDP-11, or IBM computer. Many software professionals are switching from flowcharts to a PDL, for two main reasons: First, too many people write a flowchart after the program is done, producing nothing more than the instructions with a block around them. Second, in a production environment, the programmer does not control flowcharts after sketching them. They are sent to the drafting department to be readied for publication. Design changes rarely catch up to the original documentation.

Repeat...Until

One PDL has the following instructions:

```
Repeat
            program statements
                    "
```

```
Until condition
```

The loop of instructions between the Repeat and Until is repeated until the condition is satisfied. The condition specified is checked at the end of each pass through the loop. When the loop is finished, control moves to the line of code following the Until statement. Use this construction for loops which must repeat at least once.

Example: Design a routine that will read a tape and store the contents in memory until the tape is finished or the buffer is full.

```
Repeat
  Get character from tape
  Store in buffer
  Increment buffer pointer
Until (buffer full) Or (End of tape)
Display finished prompt
```

While...Do

The second set of instructions has the form:

```
While condition Do
      program statement(s)
End While
```

This instruction is similar to the Repeat except that the check Of condition is made before the loop Of is executed. When the condition is satisfied, control passes to the instruction following the End While. If the condition is false initially, the loop will not be executed.

Example: Design a program to read a disk file and store data in buffer until the end of file is found.

```
While Not EOF Do
  Get sector from disk
  Store in buffer
  Increment buffer pointer
End While
```

If...Then...Else

The third set of instructions has the form:

```
If condition
  Then program segment 1
  Else program segment 2
```

This is identical to the If...Then...Else from Basic, Fortran, or Pascal. Only one of the Then or Else program segments is executed. At the completion of the path taken, control transfers to the instruction following the Else clause. The Else clause is usually optional.

Example: Design a program to read a tape until the end of tape is found, the buffer is full, or the read is aborted by the break key.

```
stop = false
While stop = false Do
  Get character from tape
  Store in buffer
  Increment buffer pointer
  If buffer = full
    Then stop = true
    Else If EOT
      Then stop = true
      Else
        Scan keyboard
        If break key closed
          Then stop = true
End While
```

The indentations show the reader what instructions are at the same level. Each successive indentation means that the line is subordinate to the last line. This method is, however, cumbersome.

"Each successive indentation means that the line is subordinate to the last line."

Case

To overcome this, the Case statement can be used.

```
Case of
  AA: program statement
  BB: program statement
  Others: program statement
End Case
```

In the Case statement only one of the case labels (AA:, BB:, or Others) is executed. Upon completion of a particular case, control passes to the instruction following the End Case.

Example: Repeat the design of example three using the Case statement.

```
stop = false
While stop = false
Get character from tape
Case of
  End Of Tape: stop = true
    Buffer full: stop = true
       Others: store character in buffer
               scan keyboard
               If Break key
                  Then stop = true
  End Case
End While
```

This instruction clarifies nested If...Then...Else statements. The reader has a more accurate picture of logic at a glance.

Subroutines

Subroutines are identified by a name. In the above example, "store character in buffer" and "scan keyboard" are subroutines. They can be called by name (as above) or by preceding the name with the word Call.

Two schools of thought dictate where subroutines are placed in a program. The first method is to place them at the beginning of a program. Many languages (like Pascal) require this predefinition.

The second school is that they belong at the end of the program, following the concept of top-down design. Controlling modules come first, with subordinate modules following. In some languages (like Basic) the placement makes no difference to the program. But with an interpreter, programs execute faster with subroutines at the front of a program. This is because subroutines are called by line number; the interpreter must scan from the front of the program until the required line number is found.

GOTO

The GOTO label, identical to the Basic GOTO, can be either a number or word. When designing a program this is seldom used. In some cases it is more efficient, but use it sparingly.

This completes the instruction set of most PDLs. The only item remaining is a comment line. This is usually defined as: /* comment */.

In the above examples, English is used liberally to indicate desired functions. This makes reading and comprehending the program very easy, and the program transportable across languages and computers. When implementing the program, these instructions are changed to some corresponding represention in the particular language. language.

Documentation

This is the last and most important area to cover. Why are documentation and a logical approach so important? Most programs are written based on criteria valid at the time of design. As time passes requirements will change, and you will have to modify the program. If the program is poorly documented, you will not know why you did things the way you did. The documentation helps you sift through the maze of instructions making up the program. In many cases it would be easier to start from scratch and redesign the program.

How is a program documented?

There are three ways of maintaining a program documentation package. The first is the program maintenance package. The program is documented both internally and externally. Internal to the program, use simple structure (no fancy coding).

Use meaningful names and labels, define them at the beginning of the program, and initialize them. Separate variable RAM, I/O and parameter with a blank line. Use names for I/O devices and parameters definitions. Comment liberally. Preface each section with a brief description of its function. Comment as many lines as possible, but don't comment the obvious, by restating the meaning of an instruction. Include a header on each program containing the program's name, creation date, author's name, last revision date, and the name of the person modifying it.

External to the program include the engineering notebook described earlier. Describe the program logic flow and functions using flowcharts or a PDL.

The second way of maintaining a program documentation package is to write a library user's package, for documentation of subroutines. The library user's package should contain the name of the subroutine (creation date and last revision date); the purpose of routine; the computer system required; the language it's written in; the required parameters and how they are passed; the results and how they are passed; the affected registers; and error handling provisions.

This data should be kept at the beginning of each subroutine, algorithm, or module. It should also be kept in a library file, on the computer or on index cards. This way a programmer will not reinvent work someone has already done.

The third way of maintaining a program documentation package is to write a user's manual. The user's manual should include a description of the system configuration required, the amount of memory, number of disks or tape recorders, and so on; a description of the program's function, what it does, and any special functions it can perform; and any warnings or special considerations. Define any external information used. If data is *not* keyed in, include a description of data structure and required I/O. Describe the loading and execution procedure.

Solving a Problem

Let's put these tools to work writing a program to convert an Assembly file to an ASCII file.

"Let's put these tools to work writing a program to convert an Assembly file to an ASCII file."

Step one is the initial problem analysis. What do we know about the problem? We know that it is supposed to input an Assembly file, and output an ASCII file. Organize this information.

The Assembly file is the given information and the ASCII file is what the programmer is required to find.

Are there any implied assumptions?

The answer is *probably*. What assumptions were made? The first was the location at which the source file is stored. By my definition, it is on a disk. Great! That leads to two other questions: What does an Assembler source file look like, and what does an ASCII file look like on the disk? Both of these questions require further research.

This research was accomplished using several tools: Miosys Editor/Assembler to generate the input source file; Basic to generate the output ASCII file; and Apparat's Superzap to examine both the Assembler and a Basic ASCII file. A program, written with the Editor/Assembler (content and executability is not important), was saved to disk. Then a short, meaningless Basic program was written and saved to disk using the ASCII format command, Save "file name",A. Finally, Superzap was loaded, and these two files were examined using its DFS command. The format of the Assembly source file was: D3(H), a program type identifier, is the first byte; next, 6-byte file name in ASCII with bit 8 set low; third, 5-digit ASCII line number with bit 8 set high; next, N number of ASCII characters for the line with a 0D(H) terminator; and finally a 1A(H) End Of Text terminator at the end of the file. The third and fourth steps repeated until the End of Text (EOT = 1A) indicator was found on the last sector.

The format of the Basic ASCII file was: A five-digit ASCII line number with bit 8 set low; N number of ASCII characters for the line with a 0D(H) terminator; and a 1A(H) EOT terminator. This means our program can be considered a black box, with one in-

```
                    /*  CONMLS  */

/*  This program will convert an Assembly source file          */
/*  to an ASCII source file, and save it on disk.              */
/*  It can also go the other way, converting an ASCII          */
/*  file to an Assembly source file.                           */

    Initialize any data required
/*  This section will also be responsible for getting          */
/*  the source and destination file names.                     */

/*  Handle first record of file                                */
    Get sector from disk
    Call Strip /* strip off file identifier and name           */
    Output modified line to new disk file

/*  Handle remaining records except the last                   */

    While not End Of File Do
      Get sector from disk
      Call Number 2 /* convert line numbers from               */
                    /* bit 8 high to low                       */
    End While

/*  EOF character found, handle last case                      */

    Call Close /* put end of file marker on file               */

End Program
```

Table 2. Example PDL

Table 3. Initialization PDL

```
/*                    Initialization Section                        */

   Define all variables
   Display explanatory text
   Prompt user for type of input file
   Open appropriate disk files
End Initialization

   Case Of input file type
      ASCII : /*Convert to Assembly source                          */

   /*  Handle first record of file                                  */
   Get sector from disk
   Call Add /* add file identifier and name                         */
   Call Number 1   /* convert line numbers from bit 8               */
                   /* low to high                                   */
   Output modified line to new disk file

/*  Handle remaining records except the last                        */
   While not End Of File Do
      Get sector from disk
      Call Number 1
   End While

/*  EOF character found, handle last case                           */
   Close files /* put end of file marker (1AH) on file              */

         ASSBLY: /* convert to ASCII file                           */

      /*  Handle 1st record of file                                 */
   Get sector from disk
```

Table 3 continues

put and one output. The program appears to be a transfer function: It transforms the input into the desired output.

Step two is to formulate a general model. We have to re-analyze this data and identify the items to be transformed and any particular order required. Once this has been accomplished the model can be built. Most programs have two things in common. At the initial start they all can be represented by:

```
Initialization section
Input data
Solve problem
Output results
```

Next, "solve the problem" is examined. Again most programs have the following in common:

```
Process first item
Process all remaining items, except
the last
Process the last item
```

Usually there is something different about the first and last data processed. Consequently they must be handled separately.

Stating the changes that have to be

Table 3 continued

```
   Call Strip /* Strip off file identifier and name                         */
   Call Number 2 /* convert line numbers from bit 8                         */
                  /* high to low                                            */
   Output modified line to new disk file

/* Handle remaining records except the last                                 */
   While not End Of File Do
      Get sector from disk
      Call Number 2
   End While

/* EOF character found, handle last case                                    */
   Close Files /* put end of file marker (0DH) on file                      */

   End ASSBLY

      Others : Error and start again
End Case
```

made to go from an Assembly source to an ASCII file formally, the first change is to delete the file type identifier and the file name. Next, change the bit 8 high ASCII line number to bit 8 low ASCII. There is no need to change the N ASCII characters composing a line of the program.

The PDL for the first crude model would look like Table 2.

Does this PDL look accurate? Can you find any flaws? If the answers are yes and no respectively, it is time to enhance the program by writing the subordinate sections.

The PDL for the Initialize section would look like Table 3.

Check the model. If there are no errors, begin coding.

At this time the language used is defined (Basic) and coding begins. The final outcome is shown in the Program Listing. One thing should be brought to your attention again: the *necessity* of documenting a program as it is written. Remember, each section should have a description of what it does.

> *". . . the student is expected to learn programming design by default."*

Finally, execute the program and see if it works. The one in the Program Listing does. If it didn't the necessary changes should be made. If the errors are syntax errors, no correction to the flowcharts or PDL is required. If the program does not work for some reason other than syntax, there is a logic flaw; go back to the PDL and check the logic. If it appears correct, check your program to see that it matches the PDL. ■

John Blair (122 Dumont Ave., Norfolk, VA 23505) is an electrical engineer and amateur radio operator.


NEW!
ESF III
NOW AVAILABLE!
SIMPLY AMAZING!
NEW LOW PRICE—$99.50
How else can you describe the *Exatron Stringy Floppy* system? You could say that it's an under $100, compact, reliable, robust, high-speed, computer-controlled, easy-to-use, well-supported alternative to disk drives, for a Model I TRS-80—*simply amazing!*
Amazing Technology
Based on a special endless-loop tape cartridge, called a Wafer, the ESF system was designed specifically for computer data storage. The direct-drive transport mechanism has only one moving part, and data is transferred to and from the tape at a rate of 7200 baud.
Amazing System
Thousands of ESF buyers have been amazed by 16K programs loading in less than 20 seconds; automatic verification of saved programs; up to 70K bytes, and 99 files, on a single Wafer; a ROM operating system (RAM based in Model I) no need for an expansion interface; and 1-year parts and labor warranty.
Amazing Support
With an ESF system you don't just get a piece of hardware, you get total support with hundreds of user workshops; dozens of high-quality, reasonably priced programs (such as Electric Pencil 2.0, Electric Spreadsheet, File Management System and Technical Word Processor); access to hundreds of FREE public-domain programs; an @NEWS user column in 80-US; @LOAD program magazine; and a toll-free information line.
Amaze Yourself
To see for yourself how amazing the ESF system is, or for more detailed information, call us toll-free at **800-538-8559** (inside California **408-737-7111**) and take advantage of our 30-day money-back return policy. Copies of the 80-page manual are available for $4.95 (which you can credit towards an ESF), and while you're on the line ask about our equally amazing 64K RAM/ROM board for the Model I.
exatron
STRINGY FLOPPY
excellence in electronics
exatron
93
181 Commercial Street,
Sunnyvale, CA 94086
TRS-80 is a trademark of Tandy Corporat
Stringy Floppy is a trademark of Exatron Corp

PLEASE ANSWER THESE 5 QUESTIONS

	YES	NO
1. Do you have information that must be kept organized and accessible?	☐	☐
2. Do you ever need to perform statistical analyses?	☐	☐
3. Would you like to have a tool that will allow you greater flexibility in managing your own or your company's money?	☐	
4. Could you use a mail list program that will be easy to use, maintain an unlimited number of names, allow you great sorting flexibility, and even interact with a word processor?		
5. Do you own or have access to a TRS-80 microcomputer?		

If you answered "yes" to any of these questions, we can be of assistance to you. Our Maxi Series of applications programs are designed to give you maximum versatility with a minimum of hassle. These programs were created for business use, but you'll find yourself using them for personal applications as well.

Thorough support *— Of course, each program comes with in-depth, user-oriented documentation, and is menu-driven to make it easy to use. When necessary, the Maxi programs are compatible with each other, and, whenever pertinent, are interactive with the major word processing and spreadsheet programs published by other manufacturers. Also, we maintain a telephone support line to provide you with any assistance you might require.*

Maxi Manager *by Dale Kubler*

Maxi Manager is a remarkable data base manager. Its fast machine language sort complements its large data storage capacity. The sophistication of its data entry, management, and printing capabilities makes Maxi Manager a versatile tool for many applications.

The program now includes Maxi Utility, which allows you to rescue files on diskettes that have been damaged by excessive wear or misuse and lets you expand, add, or delete fields from an existing data base. Let the unmatched capabilities of Maxi Manager handle your data management!

Model I & Model III. Minimum 1 disk drive required 012-0196 $149.99

Maxi Cras *by Dale Kubler*

(Check Register Accounting System)

Maxi Cras is a system that will computerize check writing, recording and analysis for business and personal finance. The system features 223 income and expense accounts, each of which will handle an unlimited number of transactions. Extensive register and report printing capabilities make Maxi CRAS an indispensible tool for managing money effectively.

Model I & Model III. Minimum 2 drives required 012-0145 $99.95

Maxi Mail *by Dale Kubler*

Maxi Mail is a powerful mail list management system that:

1. Is easy to use.
2. Has virtually unlimited storage capacity.
3. Interfaces with the major word processors to generate form letters and other text.
4. Prints mailing labels up to four-across in any format desired.
5. Has virtually unlimited coding capabilities with thirteen fields of information for each record.

Maxi Mail is the most sophisticated user-oriented mail list program available.

TRS-80 Model III only. 2 disk drives required 012-0148 $99.95

Maxi Stat *by David Walonick*

Maxi Stat is the most useful statistical analysis package on the market today. It was developed to allow maximum flexibility in designing customized analysis. Maxi Stat handles the three main components of statistical analysis:

1. Complete menu-driven codebook creation and editing.
2. Menu-assisted data entry.
3. User-created control files to describe the statistical analyses to be performed and printed out on the variables of your choice.

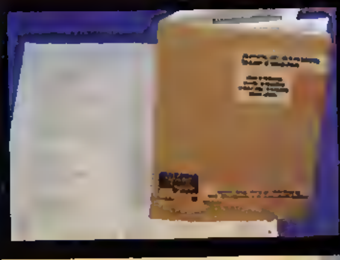
Model I & Model III. Minimum 2 drives required 012-0153 $199.95

A Division of Scott Adams, Inc.

THE BUSINESS DIVISION

Send $1.00 for our 16 page booklet "Getting The Most From Your Micro" All 16 pages are packed with indepth explanations and printout samples from the Maxi Series of applications programs.

THE BUSINESS DIVISION
BOX 3435
LONGWOOD, FL 32750
(305) 830-8194



~20

"MAXI STAT Is The Most Versatile Statistical Analysis Program Available . . . On Any Micro."

Dr. Steven E. Mayer, PhD., Industrial Psychologist — Maxi User

Focus On: Maxi Stat

Who can use Maxi Stat?

Anyone who needs an analysis of statistical data. Maxi Stat has successfully replaced dedicated mainframe and time-sharing statistics programs for business, financial and marketing researchers, educational administrators, hospitals, medical schools and medical research facilities, agricultural testing, and social scientists.

What does it do?

If you're familiar with SPSS (Statistical Package for the Social Sciences) and what it can do on a mainframe, you have a good idea of what Maxi Stat will do for your TRS-80.

Maxi Stat offers many of the best features of SPSS (modified for microcomputers) as well as features you can't get in any other system, at any price. Maxi Stat is one of the most useful analysis packages available — anywhere.

Read on to discover what this remarkable new tool can do for you . . .

Maxi Stat offers:

- menu-driven operation
- user-created codebook
- the ability to handle up to 255 variables
- machine language code allowing speedy data entry
- fast and easy editing
- hundreds of tasks performed at one time
- reports you can custom design for your specific needs
- ample screen prompts to guide user
- frequent automatic backup of data to help protect against power failures and system crashes
- all phases of research analysis, including:
  - write subfiles
  - frequency distribution
  - descriptive statistics
  - crosstabs & chi-square
  - correlation and linear regression
  - t-test
  - multiple linear regression
  - analysis of variance
  - multiple variable response

Summary

See for yourself what hundreds of other professionals have already discovered — Maxi Stat is THE solution to your analysis problems, and it's never further away than your TRS-80. Pull the plug on your time-sharing system and discover the power and convenience of Maxi Stat today!

THE BUSINESS DIVISION

By David Walonick

Maxi Stat is available now for TRS-80 Models I and III with 2 disk drives, 48K RAM, and printer.

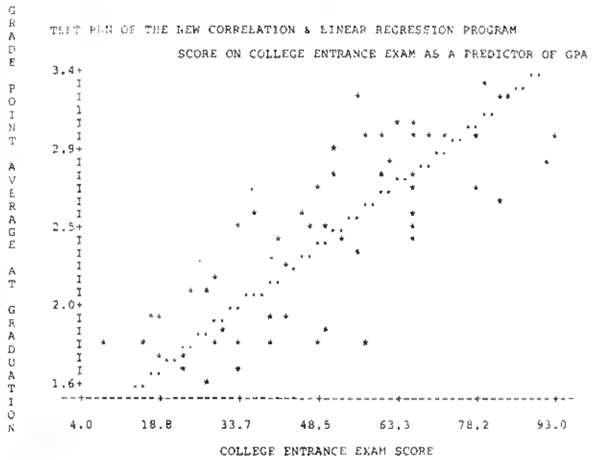
$199.95

Soon to be available for IBM PC

Look who's already using it!

University of Alaska, Fairbanks, AK. • **Florida Hospital**, Orlando, FL. • **Capital Research Services**, Topeka, KA. • **Center for Financial Studies and Development**, Claremont, CA. • **U.S. Army Psychological Testing Unit**, Wahiawa, HI. • **University of Nottingham Dept. of Social Administration**, Nottingham, U.K. • **Defense Intelligence School**, Washington, D.C. • **Dept. of Recreation and Park Administration, University of Missouri**, Columbia, MO. • **Children's Hospital of Philadelphia**, Philadelphia, PA. • **Kingston City Schools**, Kingston, N.Y. • **Universidad de Monterrey**, Monterrey, Mexico • **University of Michigan, Dept. of Pediatrics**, Ann Arbor, MI. • **Kodak Camera (Australia, Asia)**, Coburg, Victoria, Australia • **The Williams Group Advertising and P.R.**, Dallas, TX. • **Medical College of Pennsylvania**, Philadelphia, PA. • **Parkway Day School**, Philadelphia, PA. • **University of Illinois**, Chicago, IL. • **Selkirk Health District**, Nelson, B.C. Canada • **Human Communications Systems**, Reston, VA. • **California Research Center**, Los Angeles, CA. • **WaltRich Marketing Consultants**, Orlando, FL. • **Rutgers University**, Piscataway, N.J. • **Bethel Theological Seminary**, Brooklyn Park, MN • **Seton Hall University, Stillman School of Business**, S. Orange, N.J. • **Plan-Test Associates**, Phoenix, AZ. • **General Mills Corporation**, Minneapolis, MN. • **Shearson, Hayden Stone Investments**, Hollywood, CA. • **Energy Systems Research Institute**, Raleigh, N.C. • **Minnesota Historical Society**, St. Paul, MN. • **Purdue University**, W. Lafayette, ID. • **Nova Scotia Agricultural College**, Truro, N.S., Canada • **Agrico Chemical Co.**, Wilmington, DL. • **Marketing Associates**, Charlotte, N.C. • **University of Ottawa, School of Psychology**, Ottawa, Ont., Canada • **Rock House Publishing**, Kowloon, Hong Kong • **Center for Applied Social Sciences**, Boston, MA. • **Blake and Dickinson Marketing Research**, Manchester, N.H. • **United Nations**, New York, N.Y., and hundreds more!

Maxi Stat prints many different report styles.




```
MEAN OF X =  48.23     CORRELATION COEFFICIENT   =  .87    VALID CASES    =  71
S.D. OF X =  23.99     DEGREES OF FREEDOM        =  69     MISSING CASES  -  4
MEAN OF Y =  2.46      SLOPE OF REGRESSION LINE  =  .02    RESPONSE %     =  94.7
S.D. OF Y =  .67       Y INTERCEPT               =  1.29

REGRESSION EQUATION :  Y' = .02 X + 1.29
STANDARD ERROR OF ESTIMATE FOR REGRESSION  = .33
STANDARD ERROR OF CORRELATION COEFFICIENT  = .12
SIGNIFICANCE OF CORRELATION COEFFICIENT    = 0.000
```

✓20

A Subsidiary of Scott Adams, Inc.

ORDER FROM YOUR FAVORITE DEALER or CALL TOLL FREE (800) 327-7172 (ORDERS ONLY PLEASE)
SHIPPING & HANDLING ARE EXTRA - PRICES SUBJECT TO CHANGE WITHOUT NOTICE
BOX 3435, LONGWOOD, FL 32750, (305) 830-8194 (QUESTIONS)

✓See List of Advertisers on Page 563

Expensive Expansion

EXPENSIVE – The *LNW System Expansion II* and built-in comes with a full 32K of 200ns RAM, RS232c 20 MA current loop serial interface. That's for starters. Next, consider our heavy gauge steel case, power indicator lamp, gold-plated connectors, FR-2 glass epoxy circuit board with solder mask and silk screen legends. Then there is the parallel printer port, screen printer port, real time clock, and extra heavy duty onboard power supply with over current protection, over voltage protection and thermal shutdown. If that's not enough then there is the floppy disk controller, guaranteed operation at a 4MHz CPU speed and our 6 month warranty. Every one of these features is *STANDARD*. This is true system expansion. You get every 'expensive' feature without spending more.

CHEAP – Our price is $399.95. Any way you compare, features or price, LNW's *System Expansion II* is the clear winner. The LNW System has been field tested for over two years with thousands of users. It works with any DOS, is 100% TRS-80 Model I compatible and it works 'right out of the box'. If there is any doubt in your mind as to whether you should buy ours or the 'other guys', just ask an LNW owner!

WE ARE #1 – Number one in price, features, reliability, performance and delivery. LNW is committed to 'expensive' features and quality at reasonable prices. LNW is committed to support, thorough documentation, and reliability.

This has made us the number one manufacturer of system expansion units and accessories for the Model I computer.

EXPANSION OPTION – 8-inch drive capability is as easy as plugging in the *LNDoubler 5/8* option*. Now you can have any combination of single- or double-density, single- or double-sided, 8"* and/or 5" disks on-line! 8-inch disk storage increased to 591,360 bytes – 77-track single-sided, double-density or 1,182,720 bytes – 77-track double-density, double-sided.

The *LNDoubler's* unique 5/8 switch allows you to boot from 5- or 8-inch system disks and it's accessible from outside the interface. The $219.95 *LNDoubler 5/8* comes with a double-density disk operating system (DOS+ 3.3.9), *complete* with BASIC and utility programs... ready to run your software.

Each of your present 40-track, single-sided 5-inch drives will store up to 184,320 bytes (formatted storage) – that's an 80% increase in storage capacity for only half the cost of just one disk drive. With three 8-inch double-density, double-sided drives your Model I will have 3.75 Megabytes of online storage – that's more storage than a Model II or Model III!

LNW Research Corp.

2620 WALNUT Tustin, CA. 92680

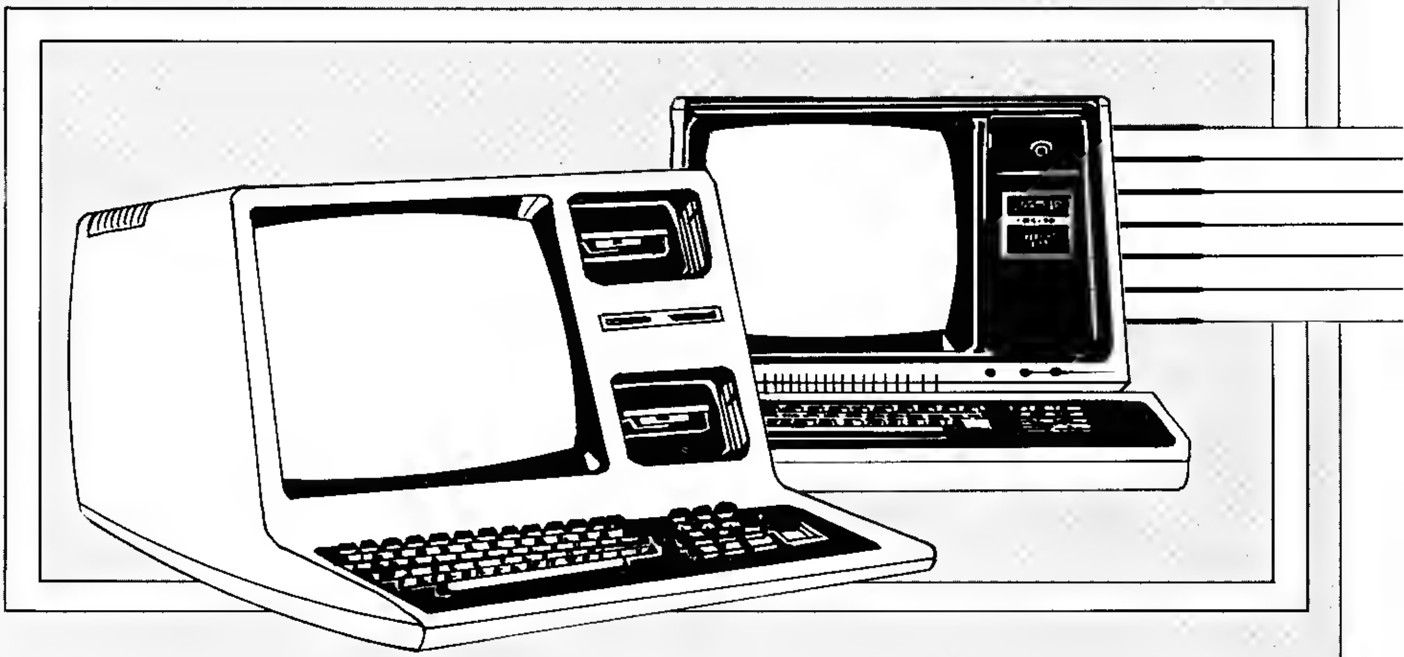
(714) 641-8850 (714) 544-5744

✓32

*8" drive operation requires special cable, 8" double-density requires 3.55MHz CPU speed-up modification or LNW-80 4MHz computer.

Part II.

MODEL I/III



NODOS 80

by Thomas Quindry

Who says cassette users can't have some of the features of disk operating systems? It certainly isn't Tom Quindry or *80 Micro*.

I've often thought it would be nice to have many of those neat Disk Basic commands and functions on a cassette-based system. There is no reason why CMD-type functions, nine USR functions, a sort routine, renumber capability, or even other customized commands and functions can't be available in a simple 12K Model I or III system.

I wanted a group of functions in memory that I could just call when needed or command from my Basic program. I decided to write an operating system for my nondisk 16K Model I.

I set a few goals for this operating system: It would be Model I/III compatible, written in Assembly language, compile to approximately 4K of machine language, reside in low memory, and all functions would be callable either by name or by a CMD-type function. Most Basic programs should run with the remaining 12K. In fact, very few Basic programs require more than 8K except for those using lots of data such as mailing lists, so losing 4K to an operating system was painless.

The question of where to begin such a monumental task is easily answered: *80 Micro*. I had all issues from the first on. I remembered reading articles describing most of the utilities I wanted.

I went through back issues to see what I could find. I felt like a kid in a candy shop with only a dollar to spend—so many possibilities but I could hardly use them all.

I eventually picked 13 articles containing Assembly-language programs and one other explaining the RST functions that I adapted for my operating system. I would have picked more, but I would have exceeded my goal of 4K.

A reference list follows this article. The articles with an asterisk were used to write NODOS 80. Additional magazine references for the same article are addenda from subsequent issues.

CMD"C":	Removes REM, ', and spaces from a Basic program.
CMD"O",A$(0):	Sorts specified string array. Specify any array.
DEFUSR:	Specify USR address. DEFUSR0–DEFUSR9 are used with USR0–USR9.
MERGE:	Merge two or more Basic programs. Used with Merge *.
RENUM:	Renumber Basic programs. Also can be called using Name.
FIND:	Locate variables in Basic programs.
PACK:	Same as CMD"C" above.
UNIKEY:	Single-key command entry. A toggle command.
DELAY:	Limited debounce routine for Unikey. Disable with Nodelay.
EDT:	Autoedit function. A toggle command.
SNGL:	Single-step through Basic. A toggle command.
DUAL:	Route video output to printer also. Same as Dual Y. Disable with Dual N.
PRDO:	Route printer output to video instead. Disabled with Dual N.
DOPR:	Route video output to printer instead. Disabled with Dual N.
&H, &O, &D:	Hexadecimal, octal, decimal conversion.

Table 1. Synopsis of Commands Available in NODOS 80

To Activate Key Codes, Press Shift Plus Alphabetical Key

A PRINT@	B ELSE	C CHR$(	D DATA
E RIGHT$(	F FOR	G GOTO	H RND(
I INPUT"	J READ	K INKEY$	L LEN(
M ASC(	N NEXT	O POKE	P PEEK(
Q LEFT$(	R RETURN	S GOSUB	T TAB(
U USING	V STRING$(	W MID$(	X SET(
Y THEN	Z RESET(		

Table 2. Unikey Single-key Entry Commands

The Key Box

Model I or III
16K RAM
Assembly Language, cassette
Editor/Assembler

See List of Advertisers on Page 563

Program Listing 1

```
                 ØØ1ØØ ;          NODOS8Ø-A        EXEC PROGRAM
                 ØØ11Ø ;          CUSTOM COMMANDS AND ERROR MESSAGES
                 ØØ12Ø ;          CMD" ", DEFUSR, USR, MERGE, SPECIAL COMMANDS
                 ØØ13Ø ;          COMPILATION, ADAPTATION OF CODE, AND NEW CODE
                 ØØ14Ø ;          BY THOMAS L. QUINDRY
                 ØØ15Ø ;
                 ØØ16Ø ;          CUSTOMIZED CMD" " FUNCTION ADAPTED FROM
                 ØØ17Ø ;          DALE W. RUPERT, 8Ø MICRO, NOV 81, P292
43ØØ             ØØ18Ø            ORG     43ØØH
43ØØ 3A7341      ØØ19Ø PATCH      LD      A,(4173H)         ;CMD
43Ø3 325843      ØØ2ØØ            LD      (STORE),A
43Ø6 2A7441      ØØ21Ø            LD      HL,(4174H)
43Ø9 225943      ØØ22Ø            LD      (STORE+1),HL
43ØC 3A5B41      ØØ23Ø            LD      A,(415BH)         ;DEFUSR
43ØF 32C643      ØØ24Ø            LD      (STORE3),A
4312 2A5C41      ØØ25Ø            LD      HL,(415CH)
4315 22C743      ØØ26Ø            LD      (STORE3+1),HL
4318 3AA941      ØØ27Ø            LD      A,(41A9H)         ;USR
431B 322C44      ØØ28Ø            LD      (STORE4),A
431E 2AAA41      ØØ29Ø            LD      HL,(41AAH)
4321 222D44      ØØ3ØØ            LD      (STORE4+1),HL
4324 3EC3        ØØ31Ø            LD      A,ØC3H            ;JUMP COMMAND
4326 327341      ØØ32Ø            LD      (4173H),A         ;CMD
4329 325B41      ØØ33Ø            LD      (415BH),A         ;DEFUSR
432C 32A941      ØØ34Ø            LD      (41A9H),A         ;USR
432F 215B43      ØØ35Ø            LD      HL,START
4332 227441      ØØ36Ø            LD      (4174H),HL
4335 21DD43      ØØ37Ø            LD      HL,START3
4338 225C41      ØØ38Ø            LD      (415CH),HL
433B 212F44      ØØ39Ø            LD      HL,START4
433E 22AA41      ØØ4ØØ            LD      (41AAH),HL
4341 Ø6ØA        ØØ41Ø            LD      B,1Ø
4343 214A1E      ØØ42Ø ERR1       LD      HL,1E4AH          ;ERROR MESSAGE
4346 22C943      ØØ43Ø ERR        LD      (DEFUS),HL
4349 214743      ØØ44Ø            LD      HL,ERR+1
434C 34          ØØ45Ø            INC     (HL)
434D 34          ØØ46Ø            INC     (HL)
434E 1ØF3        ØØ47Ø            DJNZ    ERR1
435Ø Ø614        ØØ48Ø            LD      B,2Ø
4352 35          ØØ49Ø ERR2       DEC     (HL)
4353 1ØFD        ØØ5ØØ            DJNZ    ERR2
4355 C35644      ØØ51Ø            JP      PATCH1
ØØØ3             ØØ52Ø STORE      DEFS    3
435B E5          ØØ53Ø START      PUSH    HL
435C D5          ØØ54Ø            PUSH    DE
435D F5          ØØ55Ø            PUSH    AF
435E E5          ØØ56Ø            PUSH    HL
435F 21BC43      ØØ57Ø            LD      HL,TABLE2
4362 228743      ØØ58Ø            LD      (TABLE),HL
4365 E1          ØØ59Ø            POP     HL
4366 FE22        ØØ6ØØ            CP      '"'               ;CHECK FORMAT
4368 2Ø47        ØØ61Ø            JR      NZ,QUIT
436A D7          ØØ62Ø            RST     16
436B FE41        ØØ63Ø            CP      'A'               ;BETWEEN A AND Z?
436D FAB143      ØØ64Ø            JP      M,QUIT
437Ø FE5A        ØØ65Ø            CP      'Z'
4372 CA7843      ØØ66Ø            JP      Z,OK1
4375 F2B143      ØØ67Ø            JP      P,QUIT
4378 21B643      ØØ68Ø OK1        LD      HL,TABLE1
437B BE          ØØ69Ø TEST       CP      (HL)
437C 2825        ØØ7ØØ            JR      Z,ADDR1
437E 57          ØØ71Ø            LD      D,A
437F 7E          ØØ72Ø            LD      A,(HL)
438Ø B7          ØØ73Ø            OR      A
4381 282E        ØØ74Ø            JR      Z,QUIT
4383 7A          ØØ75Ø            LD      A,D
4384 23          ØØ76Ø            INC     HL
4385 18F4        ØØ77Ø            JR      TEST
4387 ØØØØ        ØØ78Ø TABLE      DEFW    ØØØØ
4389 11B643      ØØ79Ø ADDR       LD      DE,TABLE1         ;FIND CMD" " ADDRESS
438C 3F          ØØ8ØØ            CCF
438D ED52        ØØ81Ø            SBC     HL,DE
438F 23          ØØ82Ø            INC     HL
439Ø 7D          ØØ83Ø            LD      A,L
4391 Ø7          ØØ84Ø            RLCA
4392 2A8743      ØØ85Ø            LD      HL,(TABLE)
4395 2B          ØØ86Ø            DEC     HL
4396 2B          ØØ87Ø            DEC     HL
4397 3C          ØØ88Ø            INC     A
4398 3C          ØØ89Ø            INC     A
4399 C5          ØØ9ØØ            PUSH    BC
439A 47          ØØ91Ø            LD      B,A
439B 23          ØØ92Ø INCR       INC     HL
439C 1ØFD        ØØ93Ø            DJNZ    INCR
439E C1          ØØ94Ø            POP     BC
439F 5E          ØØ95Ø            LD      E,(HL)
43AØ 23          ØØ96Ø            INC     HL
43A1 56          ØØ97Ø            LD      D,(HL)
43A2 C9          ØØ98Ø            RET
43A3 CD8943      ØØ99Ø ADDR1      CALL    ADDR
43A6 21AC43      Ø1ØØØ            LD      HL,NEXT1
43A9 E5          Ø1Ø1Ø            PUSH    HL
43AA D5          Ø1Ø2Ø            PUSH    DE                ;ADDRESS IN DE
43AB C9          Ø1Ø3Ø            RET                       ;JUMP TO "DE" ADDRESS
43AC F1          Ø1Ø4Ø NEXT1      POP     AF
43AD D1          Ø1Ø5Ø            POP     DE
43AE E1          Ø1Ø6Ø            POP     HL
43AF 23          Ø1Ø7Ø            INC     HL
43BØ C9          Ø1Ø8Ø            RET
43B1 F1          Ø1Ø9Ø QUIT       POP     AF
43B2 D1          Ø11ØØ            POP     DE
43B3 E1          Ø111Ø            POP     HL
43B4 18A2        Ø112Ø            JR      STORE
43B6 43          Ø113Ø TABLE1     DEFB    'C'     ;PACK
43B7 4F          Ø114Ø            DEFB    'O'     ;SORT
43B8 5Ø          Ø115Ø            DEFB    'P'     ;UNASSIGNED
```

Listing 1 continues

The functions I included in my operating system were CMD" "; DEFUSR; nine USR functions; a command interpreter; full error messages; hexadecimal, octal, and decimal conversion routines; a sort routine; single-step Basic, automatic editing; single-key command entry; a line-renumbering program; a program to delete spaces and remarks; a variable lister; dual and route functions for a line printer; and a new sign-on, or ready, message.

I named my operating system NODOS 80, which stands for "no disk operating system." The 80 is in honor of *80 Micro*.

The resulting program is broken down into four listings for a couple of reasons. Though the entire program compiles to approximately 4K of machine-language code, the Assembly-code listing is much longer and cannot all be compiled at once on a 16K computer. Each of the four parts fit within a 16K computer using the Radio Shack Editor/Assembler. Each can be compiled separately, and the four programs can be loaded one after the other in the order given to form the whole system.

Another reason for the spilt is so readers can customize the program to easily fit their needs. The program is modularly structured so that functions can easily be deleted or added during programming stages. You can use the reference list at the end of the article to find functions you like and want to add.

The only mandatory code is part A and part D following line 18240. Part A contains the executive program. This is the part of the program that recognizes commands. The other three parts have Equates, which only refer back to part A. Starting with line 18240 is the clean-up part of the program. This code determines where the new start of Basic will be. It sets all necessary pointers: the start of Basic pointer, the start of scaler variables, and the start and end of array variables.

Compatibility with the Model III is also maintained by code in this part. One program works on the Model I and Model III. For Model I efficiency, the program starts at 4300H. In the Model III, this starting address is in the middle of the 256-byte input buffer area. Instead of writing a new starting address for the Model III, I kept the same starting address and had the program determine if it is being used on a Model III, and if so, the input buffer is relocated to a safe area and the Basic pointers are advanced to allow for the change. Many functions also have changes from the original program to

maintain the compatibility between these two computers.

If you have no use for a particular function, just eliminate it along with all other code that refers to it. Then change the origin addresses at the start of each part following the one that was changed. The other changes will be in the origin addresses and DEFW commands listed at the bottom of parts B, C, or D.

For instance, you may already have a good renumbering program. The codes for the renumbering function of NODOS 80 take up over 25 percent of the operating-system code. Part C is entirely devoted to renumbering. If you decide you don't want the renumber function, simply ignore part C. The only change to part D would be to replace the origin address on line 13990 with 4AD3H. If you don't have a printer, the printer functions Dual and Route in lines 14000–14530 and the references to that part of the program in lines 18600–18690 can be eliminated.

If you want to add commands, that is also easy. Take, for instance, the Unikey command, to enable single-key entry commands. In part A, lines 2910–2930, have the words UNIKE, 'Y'+80H, and OUT0. The Unikey command has the last letter set with the seventh bit high (+80H). This tells the program that the end of the word to be compared has been reached. The next two bytes give the address of the Unikey function.

Notice that in part A OUT0 goes to the error-message section instead. The remark PATCH8 is where we want the program to jump to when it interprets the Unikey command. Down in part D, the program that contains Unikey, lines 18500 and 18510, fill the space in line 2930 with the proper address. This is how you add or delete commands. If a command is deleted, the program jumps to an error message as if the command name had not been recognized.

Starting on line 3210 are 31 spaces to add your own coded names and addresses to the command table. Just remember to set the seventh bit high for the last letter of the command. At this time, line 3210 is a zero byte. A zero signals the end of the command table. Be sure to start your commands here, eliminating this zero byte; after you add your last command, be sure to follow the address with a zero byte. You can add your function code anywhere before the clean-up section as long as you adjust the origin statements at the beginning of the parts B, C, or D that are affected.

Listing 1 continued

```
43B9 58        01160          DEFB    'X'     ;UNASSIGNED
43BA 5A        01170          DEFB    'Z'     ;UNASSIGNED
43BB 00        01180          DEFB    0
43BC B143      01190 TABLE2   DEFW    QUIT    ;PACK
43BE B143      01200          DEFW    QUIT    ;SABUBL
43C0 B143      01210          DEFW    QUIT    ;UNASSIGNED
43C2 B143      01220          DEFW    QUIT    ;UNASSIGNED
43C4 B143      01230          DEFW    QUIT    ;UNASSIGNED
               01240 ;        DEFUSR FUNCTION
               01250 ;        THOMAS L. QUINDRY, NEW CODE
               01260 ;        INCLUDES CODE FROM BASIC ENHANCED AGAIN
0003           01270 STORE3   DEFS    3
0014           01280 DEFUS    DEFS    20              ;DEFUSR ADDRESS TABLE
43DD F5        01290 START3   PUSH    AF
43DE D5        01300          PUSH    DE
43DF E5        01310          PUSH    HL
43E0 11C643    01320          LD      DE,STORE3
43E3 D5        01330          PUSH    DE
43E4 FEC1      01340          CP      0C1H            ;TOKENIZED USR CODE
43E6 206A      01350          JR      NZ,QUIT1
43E8 D7        01360          RST     16
43E9 FED5      01370          CP      0D5H            ;TOKENIZED EQUAL SIGN
43EB 2003      01380          JR      NZ,DIGIT1
43ED 3E30      01390          LD      A,30H
43EF 2B        01400          DEC     HL
43F0 CDF843    01410 DIGIT1   CALL    DIGIT
43F3 CD0544    01420          CALL    USR2
43F6 1820      01430          JR      DEFUS1
43F8 FE30      01440 DIGIT    CP      30H             ;BETWEEN 0 AND 9?
43FA F2FF43    01450          JP      P,OK
43FD 1853      01460          JR      QUIT1
43FF FE3A      01470 OK       CP      3AH
4401 F25244    01480          JP      P,QUIT1
4404 C9        01490          RET
4405 D62F      01500 USR2     SUB     2FH
4407 07        01510          RLCA
4408 221644    01520          LD      (STOR4A),HL
440B 21C743    01530          LD      HL,DEFUS-2
440E 47        01540          LD      B,A
440F 23        01550 INCR1    INC     HL
4410 10FD      01560          DJNZ    INCR1
4412 222544    01570          LD      (STOREH+2),HL
4415 C9        01580          RET
4416 0000      01590 STOR4A   DEFW    0000
4418 2A1644    01600 DEFUS1   LD      HL,(STOR4A)
441B D7        01610          RST     16
441C FED5      01620          CP      0D5H            ;EQUAL SIGN
441E 2032      01630          JR      NZ,QUIT1
4420 CD012B    01640          CALL    2B01H           ;EVALUATE ASCII TO BINARY
4423 ED53C943  01650 STOREH   LD      (DEFUS),DE
4427 F1        01660 USR1     POP     AF              ;RECTIFY STACK
4428 F1        01670          POP     AF
4429 D1        01680          POP     DE
442A F1        01690          POP     AF
442B C9        01700          RET
               01710 ;        USR FUNCTIONS ADAPTED FROM
               01720 ;        BASIC ENHANCED AGAIN
               01730 ;        MARK GOODWIN, 80 MICRO, NOV 81, P304
0003           01740 STORE4   DEFS    3
442F F5        01750 START4   PUSH    AF
4430 D5        01760          PUSH    DE
4431 E5        01770          PUSH    HL
4432 112C44    01780          LD      DE,STORE4
4435 D5        01790          PUSH    DE
4436 D7        01800          RST     16
4437 FE28      01810          CP      '('             ;CHECK FORMAT
4439 2003      01820          JR      NZ,DIGIT2
443B 3E30      01830          LD      A,30H
443D 2B        01840          DEC     HL
443E CDF843    01850 DIGIT2   CALL    DIGIT
4441 CD0544    01860          CALL    USR2
4444 7E        01870          LD      A,(HL)
4445 5F        01880          LD      E,A
4446 23        01890          INC     HL
4447 7E        01900          LD      A,(HL)
4448 57        01910          LD      D,A
4449 ED538E40  01920          LD      (408EH),DE
444D 2A1644    01930          LD      HL,(STOR4A)
4450 18D5      01940          JR      USR1
4452 E1        01950 QUIT1    POP     HL              ;ERROR OCCURRED
4453 D1        01960          POP     DE
4454 F1        01970          POP     AF
4455 C9        01980          RET
               01990 ;        MERGE AND MERGE* COMMANDS ADAPTED FROM
               02000 ;        DALE W. RUPERT, 80 MICRO, NOV 81, P292
4456 215E44    02010 PATCH1   LD      HL,START1
4459 228C41    02020          LD      (418CH),HL              ;MERGE COMMAND
445C 181D      02030          JR      PATCH2
445E 7E        02040 START1   LD      A,(HL)
445F FECF      02050          CP      0CFH                    ;TOKENIZED *
4461 280A      02060          JR      Z,MERGE2
4463 2AF940    02070 MERGE    LD      HL,(40F9H)              ;END OF BASIC
4466 2B        02080          DEC     HL
4467 2B        02090          DEC     HL
4468 22A440    02100          LD      (40A4H),HL              ;START OF BASIC
446B 1806      02110          JR      BASIC
446D 2A7944    02120 MERGE2   LD      HL,(BAS)
4470 22A440    02130          LD      (40A4H),HL
4473 01181A    02140 BASIC    LD      BC,1A18H                ;MODEL I/III JUMP
4476 C3AE19    02150          JP      19AEH                   ;TO BASIC
4479 0000      02160 BAS      DEFW    0                       ;LOWEST BASIC START
               02170 ;        SPECIAL COMMAND RECOGNITION ADAPTED FROM
               02180 ;        WHITE
               02190 ;        JAKE COMMANDER, 80 MICRO, JULY 80, P94
447B 3EC3      02200 PATCH2   LD      A,0C3H
447D 32A641    02210          LD      (41A6H),A               ;ERROR
4480 218944    02220          LD      HL,ENTRY
4483 22A741    02230          LD      (41A7H),HL
4486 C33D47    02240          JP      PATCH3
4489 F5        02250 ENTRY    PUSH    AF
```

Listing 1 continues

The command tables for CMD" " are similar, except that only single-letter commands are used. In this case, TABLE1 corresponds with the addresses in TABLE2 (lines 1130 -1230). Extra unassigned commands P, X, and Z are given. You can change them to suit your needs, or you can use them by putting the correct function addresses in lines 1210, 1220, and 1230, respectively. Notice the one-for-one correspondence between TABLE1 and TABLE2 as given by the remarks.

Table 1 gives a synopsis of all of the functions. The rest of this article gives more detail on how to use them.

CMD" " Function

Two commands are available. First, CMD"C" calls the pack function. This function compresses your Basic program in memory by removing all spaces and remark statements. It will remove both the REM and the (')-type remark statement. If a line is entirely a remark statement, the line is deleted.

Second, commands of the form CMD"O",A$(0) alphabetically sort the string array specified. You can specify any string array. It is mandatory that the subscript zero array be specified as above.

It is a good idea to dimension every string array to be sorted to exactly the number of elements to be used. If the array is not dimensioned, such as DIM A$(5), a dimension of 10 is assumed. Unused strings have a null value and are sorted to the beginning of the array elements. For example, if only five strings are defined and the string array is not dimensioned, the sorted array starts with null strings from the 0–5 elements. The array elements that have been defined will start with element 6.

Specified Commands

The DEFUSR command is used to specify the address for the corresponding USR command. These commands can specify DEFUSR0–DEFUSR9 and USR0–USR9. The zero is optional. DEFUSER works exactly as it does in Disk Basic. Enter it in the form DEFUSR7 = 32000. USR also works in the same manner as in Disk Basic. Enter it in the form X = USR7(Y) or PRINT USR7(Y) where the value or variable Y can be passed to the called machine-language routine.

Merge lets you CLOAD two or more Basic programs consecutively in memory. The command temporarily hides the program already entered by redefining the start of Basic pointer to the address just above where the program

See List of Advertisers on Page 563

resides. After an additional program has been entered, the Merge * command restores the start of Basic pointer to the beginning of the first program. The space between Merge and * is optional. Use the RENUM or Name command to renumber the last program entered so that line numbers do not conflict with previous programs entered. Then use the Merge * command to restore the start-of-Basic pointer to its original position.

RENUM renumbers your Basic program in memory. The Name command provides the same function. This optional command name is given because it is similar to the renumbering function in Disk Basic.

Four options are available once you are in the renumbering program. "En-

Listing 1 continued

```
448A 7B                   LD      A,E
448B FE02                 CP      2             ;SYNTAX ERROR?
448D C25545               JP      NZ,OUT1
4490 E5                   PUSH    HL
4491 D5                   PUSH    DE
4492 2AE640               LD      HL,(40E6H)
4495 7E                   LD      A,(HL)
4496 B7                   OR      A
4497 2004                 JR      NZ,NCHR
4499 23                   INC     HL
449A 23                   INC     HL
449B 23                   INC     HL
449C 23                   INC     HL
449D D7           NCHR    RST     16
449E 11DA44               LD      DE,CMD
44A1 E5                   PUSH    HL
44A2 1A           CMPR    LD      A,(DE)        ;COMPARE WITH COMMAND TABLE
44A3 B7                   OR      A
44A4 CA5245               JP      Z,OUTØ        ;RAN OUT OF WORDS TO COMPARE

44A7 CBBF                 RES     7,A           ;GET RID OF HI BIT
44A9 BE                   CP      (HL)
44AA 2009                 JR      NZ,NEXT
44AC 1A                   LD      A,(DE)
44AD CB7F                 BIT     7,A
44AF 200F                 JR      NZ,DONE
44B1 23                   INC     HL
44B2 13           CMPR1   INC     DE
44B3 18ED                 JR      CMPR
44B5 1A           NEXT    LD      A,(DE)
44B6 CB7F                 BIT     7,A
44B8 13                   INC     DE
44B9 28FA                 JR      Z,NEXT
44BB 13                   INC     DE
44BC E1                   POP     HL
44BD E5                   PUSH    HL
44BE 18F2                 JR      CMPR1
44CØ F1           DONE    POP     AF            ;FOUND COMMAND
44C1 F1                   POP     AF
44C2 F1                   POP     AF
44C3 F1                   POP     AF
44C4 F1                   POP     AF
44C5 E5                   PUSH    HL
44C6 21D144               LD      HL,EXIT1
44C9 E5                   PUSH    HL            ;PUT "HL" ADDRESS ON STACK
44CA 13                   INC     DE
44CB 1A                   LD      A,(DE)
44CC 6F                   LD      L,A
44CD 13                   INC     DE
44CE 1A                   LD      A,(DE)
44CF 67                   LD      H,A
44DØ E9                   JP      (HL)          ;JUMP TO COMMANDED ROUTINE
44D1 AF           EXIT1   XOR     A
44D2 329A4Ø               LD      (4Ø9AH),A     ;CLEAR ERROR
44D5 E1                   POP     HL
44D6 D7                   RST     16
44D7 C31E1D               JP      1D1EH         ;BACK TO PROGRAM
44DA 52           CMD     DEFM    'RENU'        ;COMMAND TABLE
44DE CD                   DEFB    'M'+8ØH       ;END BIT + 8ØH
44DF 5245                 DEFW    OUTØ    ;START7 (ADDRESS TO GO TO)
44E1 46                   DEFM    'FIN'
44E4 C4                   DEFB    'D'+8ØH
44E5 5245                 DEFW    OUTØ    ;FIND
44E7 5Ø                   DEFM    'PAC'
44EA CB                   DEFB    'K'+8ØH
44EB 5245                 DEFW    OUTØ    ;PACK
44ED 55                   DEFM    'UNIKE'
44F2 D9                   DEFB    'Y'+8ØH
44F3 5245                 DEFW    OUTØ    ;PATCH8
44F5 44                   DEFM    'DELA'
44F9 D9                   DEFB    'Y'+8ØH
44FA 5245                 DEFW    OUTØ    ;BOUNCE
44FC 4E                   DEFM    'NODELA'
45Ø2 D9                   DEFB    'Y'+8ØH
45Ø3 5245                 DEFW    OUTØ    ;NOBOU
45Ø5 45                   DEFM    'ED'
45Ø7 D4                   DEFB    'T'+8ØH
45Ø8 5245                 DEFW    OUTØ    ;PATCHA
45ØA 53                   DEFM    'SNG'
45ØD CC                   DEFB    'L'+8ØH
45ØE 5245                 DEFW    OUTØ    ;PATCHB
451Ø 44                   DEFM    'DUAL '
4515 CE                   DEFB    'N'+8ØH
4516 5245                 DEFW    OUTØ    ;DUALOF
4518 44                   DEFM    'DUAL '
451D D9                   DEFB    'Y'+8ØH
451E 5245                 DEFW    OUTØ    ;DUALON
452Ø 44                   DEFM    'DUA'
4523 CC                   DEFB    'L'+8ØH
4524 5245                 DEFW    OUTØ    ;DUALON
```

Listing 1 continues

ter start line" and "enter end line" specify which lines you wish to renumber. "Enter new start line" specifies the line number of the first line to be renumbered. "Enter increment" specifies the difference between renumbered lines.

Beware! If you renumber lines in the middle of a program and your increment is too large to keep within the boundaries of lines not renumbered, the program may not run properly. And it may not be easily corrected if GOTOs or GOSUBs refer to those line numbers either before or after renumbering. Also, too large an increment may overflow the 65529 number limit and continue numbering with a lower number. Sometimes using RENUM gives erroneous error messages specifying nonexistent line numbers. Your program is not affected by these messages.

The Find command locates variables used in the Basic program in memory. You can specify up to three letters, including the $, #, !, and % signs. It doesn't recognize tokenized words, but to some extent it recognizes partial strings up to three letters. It recognizes three-byte combinations and one or two-byte combinations followed by a space. It does not check the variable type, so you must specify the variable exactly as it appears in the text of your program. For example, if you have a DEFINT A command, check for A and not A% if the variable appears in the program as A.

Pack provides the same function as CMD"C". The CMD"C" command was included because it is similar to the Disk Basic compress function. Some may prefer to use the Pack command instead.

Unikey allows single-key entries by hitting the shift key and any alphabetic key or the down arrow. Table 2 gives the coded entry conditions for Unikey. The shift, down-arrow combination enters a user defined command or commands. This command line is predefined as RUN<CR>. You can change it to any command or string of commands up to 64 characters by entering shift, clear. You can also enter the carriage return as part of your command. After the command line has been defined, you must enter shift, clear again to put the command in memory for later use. Unikey is a toggle command. Entering the command again disables the function. You can enable and disable it as often as you like.

Delay adds some debounce to the Unikey function. This is not needed in

Listing continued

```
4526 50       03150         DEFM    'PRD'
4529 CF       03160         DEFB    'O'+80H
452A 5245     03170         DEFW    OUT0     ;PRDO
452C 44       03180         DEFM    'DOP'
452F D2       03190         DEFB    'R'+80H
4530 5245     03200         DEFW    OUT0     ;DOPR
4532 00       03210         DEFB    0
001E          03220         DEFS    30                ;ROOM FOR MORE COMMANDS
4551 00       03230         DEFB    0
4552 D1       03240 OUT0    POP     DE                ;GIVE ERROR
4553 D1       03250         POP     DE
4554 E1       03260         POP     HL
4555 F1       03270 OUT1    POP     AF                ;GIVE ERROR MESSAGE
4556 C30F47   03280         JP      ERRMES
              03290 ;       FULL ERROR
              03300 ;       HARRY AND KEY KEAIRNS, 80 MICRO, OCT 81, P340
4559 4E       03310 NF      DEFM    'NEXT without FOR"'
456A 53       03320 SN      DEFM    'Syntax"'
4571 52       03330 RG      DEFM    'RETURN without GOSUB"'
4586 4F       03340 OD      DEFM    'Out of DATA"'
4592 49       03350 FC      DEFM    'Illegal function call"'
45A8 4F       03360 OV      DEFM    'Overflow"'
45B1 4F       03370 OM      DEFM    'Out of memory"'
45BF 55       03380 UL      DEFM    'Undefined line number"'
45D5 53       03390 BS      DEFM    'Subscript out of range"'
45EC 52       03400 DD      DEFM    'Redimensioned array"'
4600 44       03410 DO      DEFM    'Division by zero"'
4611 49       03420 ID      DEFM    'Illegal direct"'
4620 54       03430 TM      DEFM    'Type mismatch"'
462E 4F       03440 OS      DEFM    'Out of string space"'
4642 53       03450 LS      DEFM    'String too long"'
4652 53       03460 ST      DEFM    'String formula too complex"'
466D 43       03470 CN      DEFM    'Can'
4670 27       03480         DEFB    27H               ;APOSTROPHE
4671 74       03490         DEFM    't continue"'
467C 4E       03500 NR      DEFM    'No RESUME"'
4686 52       03510 RW      DEFM    'RESUME without"'
4695 55       03520 UE      DEFM    'Unprintable"'
46A1 4D       03530 MO      DEFM    'Missing operand"'
46B1 42       03540 FD      DEFM    'BAD file data"'
46BF 4E       03550 L3      DEFM    'N/A in NODOS 80 - Disk BASIC only"'
46E1 5945     03560 TABL    DEFW    NF
46E3 6A45     03570         DEFW    SN
46E5 7145     03580         DEFW    RG
46E7 8645     03590         DEFW    OD
46E9 9245     03600         DEFW    FC
46EB A845     03610         DEFW    OV
46ED B145     03620         DEFW    OM
46EF BF45     03630         DEFW    UL
46F1 D545     03640         DEFW    BS
46F3 EC45     03650         DEFW    DD
46F5 0046     03660         DEFW    DO
46F7 1146     03670         DEFW    ID
46F9 2046     03680         DEFW    TM
46FB 2E46     03690         DEFW    OS
46FD 4246     03700         DEFW    LS
46FF 5246     03710         DEFW    ST
4701 6D46     03720         DEFW    CN
4703 7C46     03730         DEFW    NR
4705 8646     03740         DEFW    RW
4707 9546     03750         DEFW    UE
4709 A146     03760         DEFW    MO
470B B146     03770         DEFW    FD
470D BF46     03780         DEFW    L3
470F 21E146   03790 ERRMES  LD      HL,TABL
4712 09       03800         ADD     HL,BC
4713 D5       03810         PUSH    DE
4714 5E       03820         LD      E,(HL)
4715 23       03830         INC     HL
4716 56       03840         LD      D,(HL)
4717 EB       03850         EX      DE,HL
4718 D1       03860         POP     DE
4719 CDA728   03870         CALL    28A7H
471C 211D19   03880         LD      HL,191DH          ;ERROR POINTER
471F E5       03890         PUSH    HL
4720 2AEA40   03900         LD      HL,(40EAH)
4723 E3       03910         EX      (SP),HL
4724 CDA728   03920         CALL    28A7H
4727 E1       03930         POP     HL
4728 11FEFF   03940         LD      DE,0FFFEH
472B DF       03950         RST     18H
472C CA7406   03960         JP      Z,0674H
472F 7C       03970         LD      A,H
4730 A5       03980         AND     L
4731 3C       03990         INC     A
4732 C4A70F   04000         CALL    NZ,0FA7H
4735 3EC1     04010         LD      A,0C1H
4737 CD8B03   04020         CALL    038BH
473A C31C1A   04030         JP      1A1CH
473D C37344   04040 PATCH3  JP      BASIC
02B2          04050         END     02B2H
00000 TOTAL ERRORS
```

Program Listing 2

```
              04060 ;       NODOS80-B        FUNCTIONS
              04070 ;       HEX/DEC/BINARY, READY MESSAGE,
              04080 ;       VARIABLE LISTER, PACKER
              04090 ;
              04100 ;       HEX/OCTAL/DECIMAL CONVERSION FROM
              04110 ;       CONSTANT ALTERNATIVES
              04120 ;       EVAN C. HAND, SR., 80 MICRO, MAR 81, P225
473D          04130         ORG     473DH
473D 3A9441   04140 PATCH3  LD      A,(4194H)         ;& FUNCTION
```

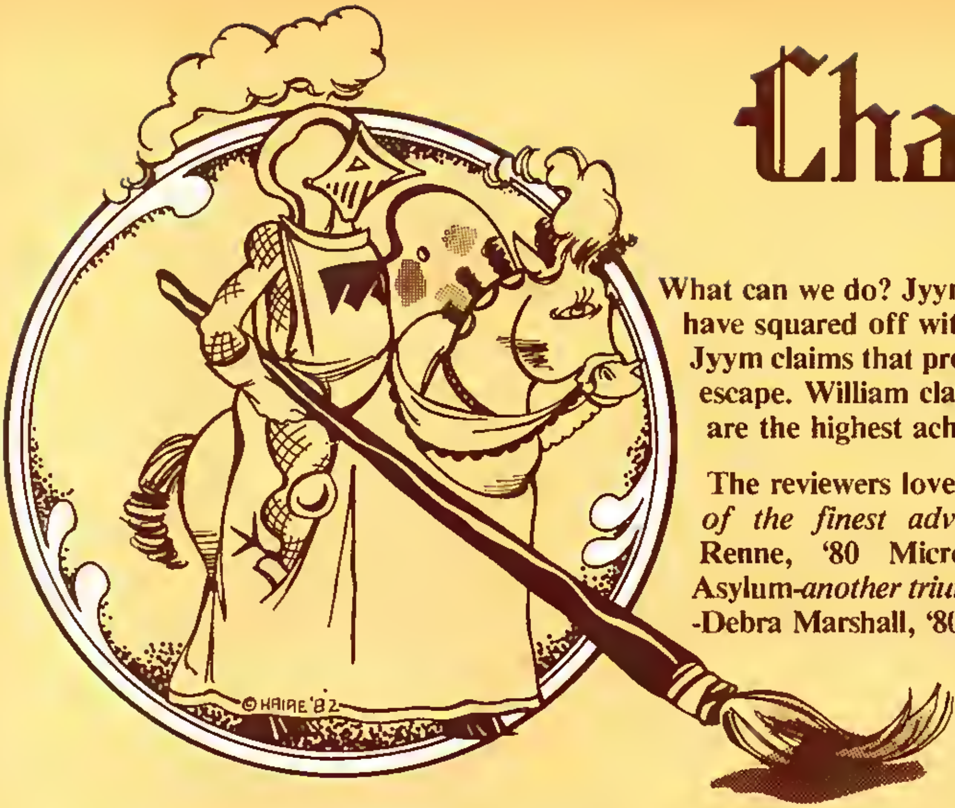
Listing continues

Listing 2 continued

```
4740 325747            LD      (STORE5),A
4743 2A9541            LD      HL,(4195H)
4746 225847            LD      (STORE5+1),HL
4749 3EC3              LD      A,0C3H
474B 329441            LD      (4194H),A
474E 215A47            LD      HL,START5
4751 229541            LD      (4195H),HL
4754 C3AF47            JP      PATCH4
0003           STORE5  DEFS    3
475A D7        START5  RST     16
475B 110000            LD      DE,0
475E FE48              CP      'H'
4760 2815              JR      Z,HEX
4762 FE4F              CP      'O'
4764 282F              JR      Z,OCT
4766 FE44              CP      'D'
4768 C25747            JP      NZ,STORE5
476B CD5E1E    DEC     CALL    1E5EH           ;ASCII TO BINARY
476E EB                EX      DE,HL
476F CD9A0A    LDINT   CALL    0A9AH           ;INTEGER LOAD
4772 EB                EX      DE,HL
4773 C9                RET
4774 C3B207    OVERR   JP      07B2H           ;OVERFLOW ERROR
4777 D7        HEX     RST     16
4778 EB                EX      DE,HL
4779 300C              JR      NC,ALPHA
477B E60F              AND     0FH
477D 0604      SHIFT   LD      B,04H
477F CDA947            CALL    SHFLFT
4782 B5                OR      L
4783 6F                LD      L,A
4784 EB                EX      DE,HL
4785 18F0              JR      HEX
4787 28E6      ALPHA   JR      Z,LDINT
4789 FE41              CP      'A'             ;BETWEEN A AND F?
478B 38E2              JR      C,LDINT
478D FE47              CP      'G'
478F 30DE              JR      NC,LDINT
4791 D637              SUB     37H
4793 18E8              JR      SHIFT
4795 D7        OCT     RST     16
4796 EB                EX      DE,HL
4797 30D6              JR      NC,LDINT
4799 FE38              CP      '8'
479B 30D7              JR      NC,OVERR
479D 0603              LD      B,03
479F CDA947            CALL    SHFLFT
47A2 E60F              AND     0FH
47A4 B5                OR      L
47A5 6F                LD      L,A
47A6 EB                EX      DE,HL
47A7 18EC              JR      OCT
47A9 29        SHFLFT  ADD     HL,HL
47AA 38C8              JR      C,OVERR
47AC 10FB              DJNZ    SHFLFT
47AE C9                RET
               ;       NODOS 80 MESSAGE FROM
               ;       NEVER READY
               ;       RON BALEWSKI, 80 MICRO, JULY 81, P199
47AF 21BD47    PATCH4  LD      HL,START6
47B2 22AD41            LD      (41ADH),HL
47B5 3EC3              LD      A,0C3H
47B7 32AC41            LD      (41ACH),A
47BA C3D34A            JP      PATCH7
47BD CDF801    START6  CALL    01F8H           ;SIMILAR TO ROM
47C0 CDF920            CALL    20F9H
47C3 21CD47            LD      HL,MESS         ;SUBSTITUTE MESSAGE
47C6 CDA728            CALL    28A7H
47C9 E1                POP     HL
47CA C32B1A            JP      1A2BH           ;RETURN TO ROM
47CD 4E        MESS    DEFM    'NODOS 80 READY'
47DB 0D                DEFB    0DH
47DC 00                DEFB    0
               ;       FIND VARIABLES IN BASIC ADAPTED FROM
               ;       LOST IN BASIC
               ;       MARK C. PAXTON, 80 MICRO, JAN 82, P304
0000           BEGWS   DEFS    0
0002           SVLINE  DEFS    2
0004           FIELD   DEFS    4
0003           HLDFLD  DEFS    3
47E6 4E        PRGMID  DEFM    'NODOS 80 Locator - Variable ='
4803 0D                DEFB    0DH
4804 4C                DEFM    'Line(s)'
480B 0D00              DEFW    000DH
480D 20        MSG1    DEFM    '            '  ;12 SPACES
4819 0D                DEFB    0DH
481A 4E                DEFM    'No Basic Program in Memory'
4834 0D00              DEFW    000DH
4836 3E5A      VALCHR  LD      A,'Z'
4838 BE                CP      (HL)
4839 FA6248            JP      M,NONVAL
483C 3E2F              LD      A,2FH
483E BE                CP      (HL)
483F F26248            JP      P,NONVAL
4842 3E39              LD      A,'9'
4844 BE                CP      (HL)
4845 F24E48            JP      P,VALID
4848 3E40              LD      A,'@'
484A BE                CP      (HL)
484B FA4E48            JP      M,VALID
484E 3E01      VALID   LD      A,1
4850 C9                RET
4851 7E        SPCCHR  LD      A,(HL)
4852 FE24              CP      '$'
4854 28F8              JR      Z,VALID
4856 FE23              CP      '#'
4858 28F4              JR      Z,VALID
485A FE21              CP      '!'
485C 28F0              JR      Z,VALID
485E FE25              CP      '%'
4860 28EC              JR      Z,VALID
4862 AF        NONVAL  XOR     A
4863 C9                RET
4864 C1        FIND    POP     BC
4865 E1                POP     HL              ;GET HL
4866 E5                PUSH    HL              ;RECTIFY STACK
4867 C5                PUSH    BC
4868 D5                PUSH    DE
4869 E5                PUSH    HL              ;SAVE HL
486A 21DD47            LD      HL,BEGWS
486D 11DE47            LD      DE,BEGWS+1
4870 010800            LD      BC,8
4873 3600              LD      (HL),00
4875 EDB0              LDIR
4877 E1                POP     HL
4878 D7                RST     10H
4879 11DF47            LD      DE,FIELD
487C 0603              LD      B,3
487E 7E        TRANS   LD      A,(HL)
487F B7                OR      A
4880 2805              JR      Z,CONT
4882 12                LD      (DE),A
4883 13                INC     DE
4884 23                INC     HL
4885 10F7              DJNZ    TRANS
4887 CDC901    CONT    CALL    01C9H
488A 21E647            LD      HL,PRGMID
488D CDA728            CALL    28A7H
4890 CD9548            CALL    CHECK1
4893 181C              JR      CONT2
4895 ED4BA440  CHECK1  LD      BC,(40A4H)
4899 2AF940            LD      HL,(40F9H)
489C 2B                DEC     HL
489D 2B                DEC     HL
489E B7                OR      A
489F ED42              SBC     HL,BC
48A1 C0                RET     NZ
48A2 21113C            LD      HL,3C11H
48A5 222040            LD      (4020H),HL
48A8 210D48            LD      HL,MSG1
48AB CDA728            CALL    28A7H
```

Listing 2 continues


RISING
STARS
MOON PATROL
VC
TELENGARD
andromeda conquest



Champions

What can we do? Jyym Pearson and William Denman have squared off with magic swords and battle axes. Jyym claims that prose adventures allow the ultimate escape. William claims that 3-D graphic adventures are the highest achievement in computer gaming.

The reviewers love both systems: The Institute-*one of the finest adventures I've ever played*-Mark Renne, '80 Microcomputing, September 1982. Asylum-*another triumphant teaser from Med Systems* -Debra Marshall, '80 Microcomputing May 1982.

Either way, you, the user, get a game that can't be solved without incredible effort. We do not exaggerate when we claim to do more in 16K than any of our competitors.

THE ASYLUM ADVENTURES

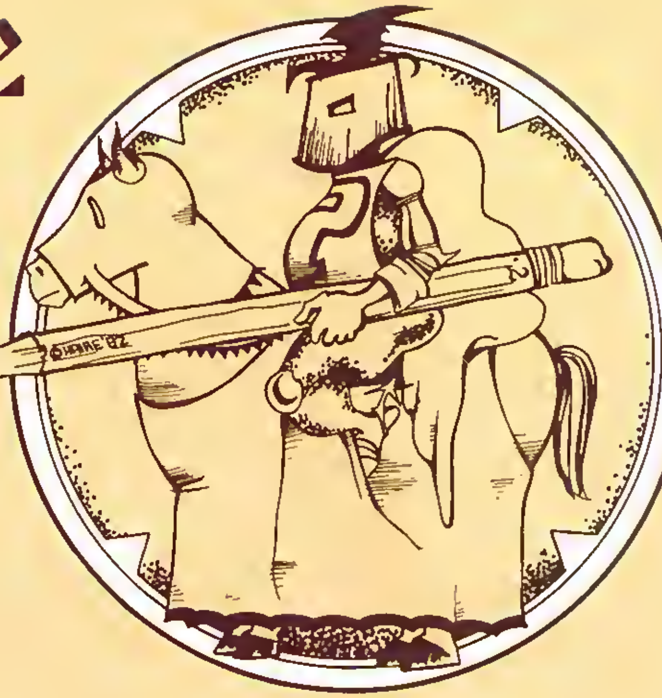
Asylum is the most sophisticated, sinister, challenging 3-D graphics adventure ever written. You are placed in an asylum full of bizarre characters, weird objects, and strange happenings. Your only goal — ESCAPE! Over 1200 locations conspire to confuse your senses. Hallways recede into the screen as though you are actually there. An advanced language interpreter understands complete English sentences, not just one and two word commands. And everything takes place instantly!

Asylum 16K cassette or diskette $19.95

Just when you thought it was safe to go back to the keyboard...Asylum II was born! The insanity continues. You need not have played Asylum I to play Asylum II. You will need ingenuity, persistence, and a terrific sense of direction to negotiate the 1500 locations full of army ants, rats, and strange inmates.

Asylum II 16K cassette or diskette $19.95

# of Adventure



Our cassettes are reliable, and our disks feature the Brandon loader. This allows multiple game saves and immediate compatibility with all disk systems (80 track [96 tpi only], LNW, Model I double density, or any other TRS-80 configuration you care to name). You don't have to be a systems programmer to boot a Med Systems disk. One disk fits all systems.

Still not convinced? Then remember, Med Systems offers a FULL MONEY BACK GUARANTEE! If you don't like our product, return it within 14 days for a prompt refund. Ask our competitors if they will do that!

THE PEARSON ADVENTURES

The Institute: not just one adventure, but five! You must negotiate not only the institution, but four other scenarios in your dreams. A huge statue, a prehistoric forest, an ancient temple, and the Titanic ocean liner all conceal objects and clues to help you escape the nightmare of the Institute.

The Institute 16K cassette or diskette $19.95

Lucifer's Realm: Entering the kingdom of Satan, you discover a revolution in the making, headed by Adolf Hitler. Travel through the bowels of Hades, dealing with the most evil mortals of all time. By cunning and strategy, you can bring Hitler to his final doom and gain your escape from the fiery Pit.

Lucifer's Realm 16K cassette or diskette $19.95

The Paradise Threat: Successfully emerging from Lucifer's Realm, you find that even heaven is not immune from harm. Your own previous actions in hell have led to a threat within heaven itself. Winston Churchill, Abe Lincoln, and Groucho Marx guide you to ways to remedy your mistakes and save Paradise.

Paradise Threat 16K cassette or diskette $19.95

The Farvar Legacy: You inherit an ancient European castle, only to discover that it is not empty. But its inhabitants, including some of your ancestors, are "the undead." Battle these terrifying creatures through the musty halls of the castle in order to save your own immortal soul.

The Farvar Legacy 16K cassette or diskette $19.95

Med Systems Software • P.O. Box 3558 • Chapel Hill, NC 27514
To order, call 1-800-334-5470, or see your dealer

Please add $2.00 to all orders for shipping and handling

Listing 2 continued

```
48AE C37344   05660        JP    BASIC
48B1 E5       05670 CONT2  PUSH  HL
48B2 111E3C   05680        LD    DE,3C1EH
48B5 ED532040 05690        LD    (4020H),DE
48B9 21DF47   05700        LD    HL,FIELD
48BC CDA728   05710        CALL  28A7H
48BF 11483C   05720        LD    DE,3C48H
48C2 ED532040 05730        LD    (4020H),DE
48C6 C1       05740        POP   BC
48C7 2AA440   05750        LD    HL,(40A4H)
48CA 23       05760 SAVELN INC   HL
48CB 0B       05770        DEC   BC
48CC 23       05780        INC   HL
48CD 0B       05790        DEC   BC
48CE 7E       05800        LD    A,(HL)
48CF 32DD47   05810        LD    (SVLINE),A
48D2 23       05820        INC   HL
48D3 0B       05830        DEC   BC
48D4 7E       05840        LD    A,(HL)
48D5 32DE47   05850        LD    (SVLINE+1),A
48D8 23       05860        INC   HL
48D9 0B       05870        DEC   BC
48DA AF       05880 EXAMIN XOR   A
48DB 32E347   05890        LD    (HLDFLD),A
48DE 32E447   05900        LD    (HLDFLD+1),A
48E1 32E547   05910        LD    (HLDFLD+2),A
48E4 BE       05920        CP    (HL)
48E5 2009     05930        JR    NZ,EXAM01
48E7 23       05940 BMPSAV INC   HL
48E8 0B       05950        DEC   BC
48E9 78       05960        LD    A,B
48EA B1       05970        OR    C
48EB CA8E49   05980        JP    Z,ENDPRG
48EE 18DA     05990        JR    SAVELN
48F0 CD3648   06000 EXAM01 CALL  VALCHR
48F3 FE01     06010        CP    1
48F5 2809     06020        JR    Z,EXAM02
48F7 23       06030 BMPRTN INC   HL
48F8 0B       06040        DEC   BC
48F9 78       06050        LD    A,B
48FA B1       06060        OR    C
48FB CA8E49   06070        JP    Z,ENDPRG
48FE 18DA     06080        JR    EXAMIN
4900 7E       06090 EXAM02 LD    A,(HL)
4901 32E347   06100        LD    (HLDFLD),A
4904 23       06110        INC   HL
4905 0B       06120        DEC   BC
4906 78       06130        LD    A,B
4907 B1       06140        OR    C
4908 CA8E49   06150        JP    Z,ENDPRG
490B AF       06160        XOR   A
490C BE       06170        CP    (HL)
490D 2834     06180        JR    Z,CHECK
490F CD3648   06190        CALL  VALCHR
4912 FE01     06200        CP    1
4914 280D     06210        JR    Z,EXAM03
4916 CD5148   06220        CALL  SPCCHR
4919 FE01     06230        CP    1
491B 2026     06240        JR    NZ,CHECK
491D 7E       06250        LD    A,(HL)
491E 32E447   06260        LD    (HLDFLD+1),A
4921 1820     06270        JR    CHECK
4923 7E       06280 EXAM03 LD    A,(HL)
4924 32E447   06290        LD    (HLDFLD+1),A
4927 23       06300        INC   HL
4928 0B       06310        DEC   BC
4929 78       06320        LD    A,B
492A B1       06330        OR    C
492B 2861     06340        JR    Z,ENDPRG
492D AF       06350        XOR   A
492E BE       06360        CP    (HL)
492F 2812     06370        JR    Z,CHECK
4931 CD3648   06380        CALL  VALCHR
4934 FE01     06390        CP    1
4936 2807     06400        JR    Z,SAVCHK
```

```
4938 CD5148   06410        CALL  SPCCHR
493B FE01     06420        CP    1
493D 2004     06430        JR    NZ,CHECK
493F 7E       06440 SAVCHK LD    A,(HL)
4940 32E547   06450        LD    (HLDFLD+2),A
4943 3ADF47   06460 CHECK  LD    A,(FIELD)
4946 DD21E347 06470        LD    IX,HLDFLD
494A DDBE00   06480        CP    (IX)
494D 2037     06490        JR    NZ,EXIT
494F 3AE047   06500        LD    A,(FIELD+1)
4952 DDBE01   06510        CP    (IX+1)
4955 202F     06520        JR    NZ,EXIT
4957 3AE147   06530        LD    A,(FIELD+2)
495A DDBE02   06540        CP    (IX+2)
495D 2027     06550        JR    NZ,EXIT
495F E5       06560 DISPLN PUSH  HL
4960 C5       06570        PUSH  BC
4961 2ADD47   06580        LD    HL,(SVLINE)
4964 CD9A0A   06590        CALL  0A9AH
4967 AF       06600        XOR   A
4968 CD3410   06610        CALL  1034H
496B B6       06620        OR    (HL)
496C CDD90F   06630        CALL  0FD9H
496F ED5B2040 06640        LD    DE,(4020H)
4973 213041   06650        LD    HL,4130H
4976 010600   06660        LD    BC,6
4979 EDB0     06670        LDIR
497B 13       06680        INC   DE
497C 13       06690        INC   DE
497D ED532040 06700        LD    (4020H),DE
4981 CD9B1D   06710        CALL  1D9BH
4984 C1       06720        POP   BC
4985 E1       06730        POP   HL
4986 AF       06740 EXIT   XOR   A
4987 BE       06750        CP    (HL)
4988 CAE748   06760        JP    Z,BMPSAV
498B C3F748   06770        JP    BMPRTN
498E 21403F   06780 ENDPRG LD    HL,3F40H
4991 222040   06790        LD    (4020H),HL
4994 C37344   06800        JP    BASIC
              06810 ;      PACKER ADAPTED FROM
              06820 ;      THE MEMORY EXPANDER
              06830 ;      TIM KENEALY, 80 MICRO, SEPT 81, P174
              06840 ;      USE CMD"C" OR PACK COMMAND
4997 318842   06850 PACK   LD    SP,4288H
499A CDC901   06860        CALL  01C9H
499D 21A24A   06870        LD    HL,TITLE
49A0 CDA728   06880        CALL  28A7H
49A3 CD9548   06890        CALL  CHECK1          ;SEE IF BASIC PROGRAM?
49A6 21B24A   06900        LD    HL,MSSG
49A9 CDA728   06910        CALL  28A7H
49AC 2AA440   06920        LD    HL,(40A4H)
49AF 22CF4A   06930        LD    (BUFF1),HL
49B2 7E       06940        LD    A,(HL)          ;NEXT BASIC STATEMENT
49B3 32D14A   06950        LD    (BUFF2),A
49B6 23       06960        INC   HL
49B7 7E       06970        LD    A,(HL)
49B8 32D24A   06980        LD    (BUFF2+1),A
49BB 2B       06990        DEC   HL
49BC ED5BA740 07000 COPY   LD    DE,(40A7H)      ;BUFFER AREA
49C0 010400   07010        LD    BC,4
49C3 EDB0     07020        LDIR
49C5 7E       07030 MORE   LD    A,(HL)
49C6 12       07040        LD    (DE),A
49C7 B7       07050        OR    A
49C8 2804     07060        JR    Z,DONE1
49CA 23       07070        INC   HL
49CB 13       07080        INC   DE
49CC 18F7     07090        JR    MORE
49CE 010000   07100 DONE1  LD    BC,0
49D1 2AA740   07110        LD    HL,(40A7H)
49D4 23       07120        INC   HL
49D5 23       07130        INC   HL
49D6 23       07140        INC   HL
49D7 23       07150        INC   HL
49D8 7E       07160        LD    A,(HL)
```

Listing 2 continues

Listing 2 continued

```
49D9 FE3A      Ø717Ø          CP      3AH
49DB 2ØØ6      Ø718Ø          JR      NZ,DONE2
49DD 3E2Ø      Ø719Ø          LD      A,2ØH          ;REMOVE COLON
49DF 77        Ø72ØØ          LD      (HL),A
49EØ 23        Ø721Ø          INC     HL             ;CHECKS REM (')
49E1 7E        Ø722Ø          LD      A,(HL)
49E2 2B        Ø723Ø          DEC     HL
49E3 FE93      Ø724Ø DONE2    CP      147
49E5 CA8A4A    Ø725Ø          JP      Z,ALLREM
49E8 7E        Ø726Ø CHECK3   LD      A,(HL)
49E9 FE88      Ø727Ø          CP      136
49EB 2816      Ø728Ø          JR      Z,DATA
49ED FE3A      Ø729Ø          CP      3AH
49EF 2817      Ø73ØØ          JR      Z,COLON
49F1 FE22      Ø731Ø          CP      22H
49F3 2822      Ø732Ø          JR      Z,QUOTE
49F5 FE2Ø      Ø733Ø          CP      2ØH
49F7 282C      Ø734Ø          JR      Z,SPACE
49F9 FE93      Ø735Ø          CP      147
49FB 284C      Ø736Ø          JR      Z,REM
49FD B7        Ø737Ø          OR      A
49FE 2852      Ø738Ø          JR      Z,FINISH
4AØØ 23        Ø739Ø          INC     HL
4AØ1 18E5      Ø74ØØ          JR      CHECK3
4AØ3 ØEØ1      Ø741Ø DATA     LD      C,1
4AØ5 23        Ø742Ø          INC     HL
4AØ6 18EØ      Ø743Ø          JR      CHECK3
4AØ8 79        Ø744Ø COLON    LD      A,C
4AØ9 FEØ1      Ø745Ø          CP      1
4AØB 2ØØ7      Ø746Ø          JR      NZ,NODATA
4AØD 78        Ø747Ø          LD      A,B
4AØE FEØ1      Ø748Ø          CP      1
4A1Ø 28Ø2      Ø749Ø          JR      Z,NODATA
4A12 ØEØØ      Ø75ØØ          LD      C,Ø
4A14 23        Ø751Ø NODATA   INC     HL
4A15 18D1      Ø752Ø          JR      CHECK3
4A17 78        Ø753Ø QUOTE    LD      A,B
4A18 FEØ1      Ø754Ø          CP      1
4A1A 28Ø4      Ø755Ø          JR      Z,SECOND
4A1C Ø6Ø1      Ø756Ø          LD      B,1
4A1E 18Ø2      Ø757Ø          JR      CONT3
4A2Ø Ø6ØØ      Ø758Ø SECOND   LD      B,Ø
4A22 23        Ø759Ø CONT3    INC     HL
4A23 18C3      Ø76ØØ          JR      CHECK3
4A25 78        Ø761Ø SPACE    LD      A,B
4A26 FEØ1      Ø762Ø          CP      1
4A28 2ØØ3      Ø763Ø          JR      NZ,OUT
4A2A 23        Ø764Ø INDATA   INC     HL
4A2B 18BB      Ø765Ø          JR      CHECK3
4A2D 79        Ø766Ø OUT      LD      A,C
4A2E FEØ1      Ø767Ø          CP      1
4A3Ø 28F8      Ø768Ø          JR      Z,INDATA
4A32 E5        Ø769Ø          PUSH    HL
4A33 E5        Ø77ØØ          PUSH    HL
4A34 DDE1      Ø771Ø          POP     IX
4A36 DD7EØØ    Ø772Ø LOOP     LD      A,(IX)
4A39 B7        Ø773Ø          OR      A
4A3A 28ØA      Ø774Ø          JR      Z,DONE3
4A3C DD7EØ1    Ø775Ø          LD      A,(IX+1)
4A3F DD77ØØ    Ø776Ø          LD      (IX),A
4A42 DD23      Ø777Ø          INC     IX
4A44 18FØ      Ø778Ø          JR      LOOP
4A46 E1        Ø779Ø DONE3    POP     HL
4A47 189F      Ø78ØØ          JR      CHECK3
4A49 2B        Ø781Ø REM      DEC     HL
4A4A 7E        Ø782Ø          LD      A,(HL)
4A4B FE3A      Ø783Ø          CP      3AH
4A4D 28FA      Ø784Ø          JR      Z,REM
4A4F 23        Ø785Ø          INC     HL
4A5Ø 36ØØ      Ø786Ø          LD      (HL),Ø
4A52 ED5BCF4A  Ø787Ø FINISH   LD      DE,(BUFF1)
4A56 2AA74Ø    Ø788Ø          LD      HL,(4ØA7H)
4A59 Ø1Ø4ØØ    Ø789Ø          LD      BC,4
4A5C EDBØ      Ø79ØØ          LDIR
4A5E 7E        Ø791Ø MORE2    LD      A,(HL)
4A5F 12        Ø792Ø          LD      (DE),A
4A6Ø B7        Ø793Ø          OR      A
4A61 28Ø4      Ø794Ø          JR      Z,FINIS
4A63 23        Ø795Ø          INC     HL
4A64 13        Ø796Ø          INC     DE
4A65 18F7      Ø797Ø          JR      MORE2
4A67 13        Ø798Ø FINIS    INC     DE
4A68 2ACF4A    Ø799Ø          LD      HL,(BUFF1)
4A6B 73        Ø8ØØØ          LD      (HL),E
4A6C 23        Ø8Ø1Ø          INC     HL
4A6D 72        Ø8Ø2Ø          LD      (HL),D
4A6E 2AD14A    Ø8Ø3Ø ETC      LD      HL,(BUFF2)
4A71 23        Ø8Ø4Ø          INC     HL
4A72 7E        Ø8Ø5Ø          LD      A,(HL)
4A73 B7        Ø8Ø6Ø          OR      A
4A74 281A      Ø8Ø7Ø          JR      Z,PRGEND
4A76 ED53CF4A  Ø8Ø8Ø          LD      (BUFF1),DE
4A7A 2AD14A    Ø8Ø9Ø          LD      HL,(BUFF2)
4A7D 7E        Ø81ØØ          LD      A,(HL)
4A7E 32D14A    Ø811Ø          LD      (BUFF2),A
4A81 23        Ø812Ø          INC     HL
4A82 7E        Ø813Ø          LD      A,(HL)
4A83 32D24A    Ø814Ø          LD      (BUFF2+1),A
4A86 2B        Ø815Ø          DEC     HL
4A87 C3BC49    Ø816Ø          JP      COPY
4A8A ED5BCF4A  Ø817Ø ALLREM   LD      DE,(BUFF1)
4A8E 18DE      Ø818Ø          JR      ETC
4A9Ø AF        Ø819Ø PRGEND   XOR     A
4A91 12        Ø82ØØ          LD      (DE),A
4A92 13        Ø821Ø          INC     DE
4A93 12        Ø822Ø          LD      (DE),A
4A94 13        Ø823Ø          INC     DE
4A95 EB        Ø824Ø          EX      DE,HL
4A96 22F94Ø    Ø825Ø          LD      (4ØF9H),HL
4A99 22FB4Ø    Ø826Ø          LD      (4ØFBH),HL
4A9C 22FD4Ø    Ø827Ø          LD      (4ØFDH),HL
4A9F C37344    Ø828Ø          JP      BASIC
4AA2 4E        Ø829Ø TITLE    DEFM    'NODOS 8Ø Packer'
4AB1 ØØ        Ø83ØØ          DEFB    Ø
4AB2 ØD        Ø831Ø MSSG     DEFB    ØDH
4AB3 44        Ø832Ø          DEFM    'Deleting Spaces and Remarks'
4ACE ØØ        Ø833Ø          DEFB    Ø
4ACF ØØØØ      Ø834Ø BUFF1    DEFW    Ø
4AD1 ØØØØ      Ø835Ø BUFF2    DEFW    Ø
4AD3 C37344    Ø836Ø PATCH7   JP      BASIC
4473           Ø837Ø BASIC    EQU     4473H          ;BASIC JUMP FROM PROG A
43BC           Ø838Ø          ORG     43BCH
43BC 9749      Ø839Ø          DEFW    PACK           ;SET COMMAND ADDRESSES
44E5           Ø84ØØ          ORG     44E5H
44E5 6448      Ø841Ø          DEFW    FIND
44EB           Ø842Ø          ORG     44EBH
44EB 9749      Ø843Ø          DEFW    PACK
Ø2B2           Ø844Ø          END     Ø2B2H          ;BACK TO SYSTEM PROMPT
ØØØØØ TOTAL ERRORS
```

Program Listing 3

```
               Ø845Ø ;         NODOS8Ø-C       RENUMBER
               Ø846Ø ;
               Ø847Ø ;         SPEEDY RENUMBERER
               Ø848Ø ;         ROBERT J. DOWD, 8Ø MICRO, MAR 81, P256
               Ø849Ø ;         USE EITHER NAME OR RENUM COMMAND
4AD3           Ø85ØØ           ORG     4AD3H
4AD3 3EC3      Ø851Ø PATCH7    LD      A,ØC3H
4AD5 328E41    Ø852Ø           LD      (418EH),A      ;NAME COMMAND
4AD8 21F54A    Ø853Ø           LD      HL,START7
4ADB 228F41    Ø854Ø           LD      (418FH),HL
4ADE C37C4F    Ø855Ø           JP      PATCHC
4AE1 4E        Ø856Ø RENUM     DEFM    'NODOS 8Ø Renumber'
```

Listing 3 continues

Telewriter-64™
the Color Computer Word Processor

- 3 display formats: 51/64/85 columns × 24 lines
- True lower case characters
- User-friendly full-screen editor
- Right justification
- Easy hyphenation
- Drives any printer
- Embedded format and control codes
- Runs in 16K, 32K, or 64K
- Menu-driven disk and cassette I/O
- No hardware modifications required

THE ORIGINAL
Simply stated, Telewriter is the most powerful word processor you can buy for the TRS-80 Color Computer. The original Telewriter has received rave reviews in every major Color Computer and TRS-80 magazine, as well as enthusiastic praise from thousands of satisfied owners. And rightly so.

The standard Color Computer display of 32 characters by 16 lines without lower case is simply inadequate for serious word processing. The checkerboard letters and tiny lines give you no feel for how your writing looks or reads. Telewriter gives the Color Computer a 51 column by 24 line screen display with *true lower case characters.* So a Telewriter screen looks like a printed page, with a good chunk of text on screen at one time. In fact, more on screen text than you'd get with Apple II, Atari, TI, Vic or TRS-80 Model III.

On top of that, the sophisticated Telewriter full-screen editor is so simple to use, it makes writing fun. With single-letter mnemonic commands, and menu-driven I/O and formatting, Telewriter surpasses all others for user friendliness and pure power.

Telewriter's chain printing feature means that the size of your text is never limited by the amount of memory you have, and Telewriter's advanced cassette handler gives you a powerful word processor without the major additional cost of a disk.

...one of the best programs for the Color Computer I have seen...
— Color Computer News, Jan. 1982

TELEWRITER-64
But now we've added more power to Telewriter. Not just bells and whistles, but major features that give you total control over your writing. We call this new supercharged version Telewriter-64. For two reasons.

64K COMPATIBLE
Telewriter-64 runs fully in any Color Computer — 16K, 32K, or 64K, with or without Extended Basic, with disk or cassette or both. It automatically configures itself to take optimum advantage of all available memory. That means that when you upgrade your memory, the Telewriter-64 text buffer grows accordingly. In a 64K cassette based system, for example, you get about 40K of memory to store text. So you don't need disk or FLEX to put all your 64K to work immediately.

64 COLUMNS (AND 85!)
Besides the original 51 column screen, Telewriter-64 now gives you 2 additional high-density displays: 64 × 24 and 85 × 24!! Both high density modes provide all the standard Telewriter editing capabilities, and you can switch instantly to any of the 3 formats with a single control key command.
The 51 × 24 display is clear and crisp on the screen. The two high density modes are more crowded and less easily readable, but they are perfect for showing you the exact layout of your printed page, *all on the screen at one time*. Compare this with cumbersome "windows" that show you only fragments at a time and don't even allow editing.

RIGHT JUSTIFICATION & HYPHENATION
One outstanding advantage of the full-width screen display is that you can now set the screen width to match the width of your printed page, so that "what you see is what you get." This makes exact alignment of columns possible and it makes hyphenation simple.
Since short lines are the reason for the large spaces often found in standard right justified text, and since hyphenation is the most effective way to eliminate short lines, Telewriter-64 can now promise you some of the best looking right justification you can get on the Color Computer.

FEATURES & SPECIFICATIONS:
Printing and formatting: Drives any printer (LPVII/VIII, DMP-100/200, Epson, Okidata, Centronics, NEC, C. Itoh, Smith-Corona, Terminet, etc).

Embedded control codes give full dynamic access to intelligent printer features like: underlining, subscript, superscript, variable font and type size, dot-graphics, etc.

Dynamic (embedded) format controls for: top, bottom, and left margins; line length, lines per page, line spacing, new page, change page numbering, conditional new page, enable/disable justification.

Menu-driven control of these parameters, as well as: pause at page bottom, page numbering, baud rate (so you can run your printer at top speed), and Epson font. "Typewriter" feature sends typed lines directly to your printer, and Direct mode sends control codes right from the keyboard. Special Epson driver simplifies use with MX-80.

Supports single and multi-line headers and automatic centering. Print or save all or any section of the text buffer. Chain print any number of files from cassette or disk.

File and I/O Features: ASCII format files — create and edit BASIC, Assembly, Pascal, and C programs, Smart Terminal files (for uploading or downloading), even text files from other word processors. Compatible with spelling checkers (like Spell 'n Fix).

Cassette verify command for sure saves. Cassette auto-retry means you type a load command only once no matter where you are in the tape.

Read in, save, partial save, and append files with disk and/or cassette. For disk: print directory with free space to screen or printer, kill and rename files, set default drive. Easily customized to the number of drives in the system.

Editing features: Fast, full-screen editor with wordwrap, block copy, block move, block delete, line delete, global search and replace (or delete), wild card search, fast auto-repeat cursor, fast scrolling, cursor up, down, right, left, begin line, end line, top of text, bottom of text; page forward, page backward, align text, tabs, choice of buff or green background, complete error protection, line counter, word counter, space left, current file name, default drive in effect, set line length on screen.

Insert or delete text anywhere on the screen without changing "modes." This fast "free-form" editor provides maximum ease of use. Everything you do appears immediately on the screen in front of you. Commands require only a single key or a single key plus CLEAR.

...truly a state of the art word processor... outstanding in every respect.
— The RAINBOW, Jan. 1982

PROFESSIONAL WORD PROCESSING
You can no longer afford to be without the power and efficiency word processing brings to everything you write. The TRS-80 Color Computer is the lowest priced micro with the capability for serious word processing. And only Telewriter-64 fully unleashes that capability.
Telewriter-64 costs $49.95 on cassette, $59.95 on disk, and comes complete with over 70 pages of well-written documentation. (The step-by-step tutorial will have your writing with Telewriter-64 in a matter of minutes.)
To order, send check or money order to:

Cognitec
704 Nob Ave.
Del Mar, CA 92014 ✓121

Or check your local software store. If you have questions, or would like to order by Visa or Mastercard, call us at (714) 755-1258 (weekdays, 8AM-4PM PST). Dealer inquiries invited.

(Add $2 for shipping. Californians add 6% state tax. Allow 2 weeks for personal checks. Send self-addressed stamped envelope for Telewriter reviews from CCN, RAINBOW, 80-Micro, 80-U.S. Telewriter owners: send SASE or call for information on upgrading to Telewriter-64. Telewriter-compatible spelling checker (Spell 'n Fix) and Smart Terminal program (Colorcom/E) also available. Call or write for more information.)

Apple II is a trademark of Apple Computer, Inc.; Atari is a trademark of Atari, Inc.; TRS-80 is a trademark of Tandy Corp; MX-80 is a trademark of Epson America, Inc.

Listing 3 continued

```
4AF2 ØØ         Ø857Ø          DEFB    Ø
4AF3 ØØØØ       Ø858Ø  ARRAY   DEFW    Ø
4AF5 2AB14Ø     Ø859Ø  START7  LD      HL,(4ØB1H)
4AF8 25         Ø86ØØ          DEC     H
4AF9 22F34A     Ø861Ø          LD      (ARRAY),HL
4AFC 24         Ø862Ø          INC     H
4AFD CDC9Ø1     Ø863Ø          CALL    Ø1C9H
4BØØ 21E14A     Ø864Ø          LD      HL,RENUM
4BØ3 CDA728     Ø865Ø          CALL    28A7H
4BØ6 CD9548     Ø866Ø          CALL    CHECK1
4BØ9 AF         Ø867Ø          XOR     A
4BØA Ø616       Ø868Ø          LD      B,22
4BØC 21E44E     Ø869Ø          LD      HL,INDEX
4BØF 77         Ø87ØØ  LOOP1   LD      (HL),A
4B1Ø 23         Ø871Ø          INC     HL
4B11 1ØFC       Ø872Ø          DJNZ    LOOP1
4B13 2F         Ø873Ø          CPL
4B14 32EC4E     Ø874Ø          LD      (ELINE),A
4B17 32ED4E     Ø875Ø          LD      (ELINE+1),A
4B1A 21F24E     Ø876Ø          LD      HL,NXTLPT
4B1D ED5BA44Ø   Ø877Ø          LD      DE,(4ØA4H)           ;START OF BASIC
4B21 73         Ø878Ø          LD      (HL),E
4B22 23         Ø879Ø          INC     HL
4B23 72         Ø88ØØ          LD      (HL),D
4B24 2AF34A     Ø881Ø          LD      HL,(ARRAY)
4B27 22E44E     Ø882Ø          LD      (INDEX),HL
4B2A 3EØA       Ø883Ø          LD      A,1Ø
4B2C 32EE4E     Ø884Ø          LD      (SLINE),A
4B2F 3EØA       Ø885Ø          LD      A,1Ø
4B31 32FØ4E     Ø886Ø          LD      (INCR3),A
4B34 2AF94Ø     Ø887Ø          LD      HL,(4ØF9H)
4B37 2B         Ø888Ø          DEC     HL
4B38 2B         Ø889Ø          DEC     HL
4B39 22F84E     Ø89ØØ          LD      (PEND),HL
4B3C CD5Ø4E     Ø891Ø          CALL    OPTION
4B3F 2AF24E     Ø892Ø  BEGIN5  LD      HL,(NXTLPT)
4B42 E5         Ø893Ø          PUSH    HL
4B43 ED5BF84E   Ø894Ø          LD      DE,(PEND)
4B47 CD994C     Ø895Ø          CALL    COMPAR
4B4A D2E14B     Ø896Ø          JP      NC,PASS2
4B4D E1         Ø897Ø          POP     HL
4B4E E5         Ø898Ø          PUSH    HL
4B4F 23         Ø899Ø          INC     HL
4B5Ø 23         Ø9ØØØ          INC     HL
4B51 22F64E     Ø9Ø1Ø          LD      (CURLPT),HL
4B54 E1         Ø9Ø2Ø          POP     HL
4B55 Ø6Ø4       Ø9Ø3Ø          LD      B,4
4B57 DD21F24E   Ø9Ø4Ø          LD      IX,NXTLPT
4B5B 7E         Ø9Ø5Ø  LOOP2   LD      A,(HL)
4B5C DD77ØØ     Ø9Ø6Ø          LD      (IX+Ø),A
4B5F 23         Ø9Ø7Ø          INC     HL
4B6Ø DD23       Ø9Ø8Ø          INC     IX
4B62 1ØF7       Ø9Ø9Ø          DJNZ    LOOP2
4B64 3AE94E     Ø91ØØ          LD      A,(GOFLAG)
4B67 B7         Ø911Ø          OR      A
4B68 2ØØE       Ø912Ø          JR      NZ,CKEND
4B6A 2AF44E     Ø913Ø          LD      HL,(CURLIN)
4B6D ED5BEA4E   Ø914Ø          LD      DE,(BLINE)
4B71 CD994C     Ø915Ø          CALL    COMPAR
4B74 281Ø       Ø916Ø          JR      Z,SAVEM
4B76 38C7       Ø917Ø          JR      C,BEGIN5
4B78 2AF44E     Ø918Ø  CKEND   LD      HL,(CURLIN)
4B7B ED5BEC4E   Ø919Ø          LD      DE,(ELINE)
4B7F CD994C     Ø92ØØ          CALL    COMPAR
4B82 28Ø2       Ø921Ø          JR      Z,SAVEM
4B84 3Ø5B       Ø922Ø          JR      NC,PASS2
4B86 AF         Ø923Ø  SAVEM   XOR     A
4B87 2F         Ø924Ø          CPL
4B88 32E94E     Ø925Ø          LD      (GOFLAG),A
4B8B 2AEE4E     Ø926Ø          LD      HL,(SLINE)
4B8E 11FAFF     Ø927Ø          LD      DE,6553Ø
4B91 CD994C     Ø928Ø          CALL    COMPAR
4B94 38Ø6       Ø929Ø          JR      C,LINEOK
4B96 CD614C     Ø93ØØ          CALL    ERROR1
4B99 C36E4C     Ø931Ø          JP      BASIC1
4B9C 2AE44E     Ø932Ø  LINEOK  LD      HL,(INDEX)
4B9F E5         Ø933Ø          PUSH    HL
4BAØ DDE1       Ø934Ø          POP     IX
4BA2 2AF44E     Ø935Ø          LD      HL,(CURLIN)
4BA5 DD74ØØ     Ø936Ø          LD      (IX+Ø),H
4BA8 DD2B       Ø937Ø          DEC     IX
4BAA DD75ØØ     Ø938Ø          LD      (IX+Ø),L
4BAD DD2B       Ø939Ø          DEC     IX
4BAF 2AEE4E     Ø94ØØ          LD      HL,(SLINE)
4BB2 DD74ØØ     Ø941Ø          LD      (IX+Ø),H
4BB5 DD2B       Ø942Ø          DEC     IX
4BB7 DD75ØØ     Ø943Ø          LD      (IX+Ø),L
4BBA DD2B       Ø944Ø          DEC     IX
4BBC DD22E44E   Ø945Ø          LD      (INDEX),IX
4BCØ 2AE64E     Ø946Ø          LD      HL,(LINECT)
4BC3 23         Ø947Ø          INC     HL
4BC4 22E64E     Ø948Ø          LD      (LINECT),HL
4BC7 2AF64E     Ø949Ø          LD      HL,(CURLPT)
4BCA E5         Ø95ØØ          PUSH    HL
4BCB DDE1       Ø951Ø          POP     IX
4BCD 2AEE4E     Ø952Ø          LD      HL,(SLINE)
4BDØ DD75ØØ     Ø953Ø          LD      (IX+Ø),L
4BD3 DD74Ø1     Ø954Ø          LD      (IX+1),H
4BD6 ED5BFØ4E   Ø955Ø          LD      DE,(INCR3)
4BDA 19         Ø956Ø          ADD     HL,DE
4BDB 22EE4E     Ø957Ø          LD      (SLINE),HL
4BDE C33F4B     Ø958Ø          JP      BEGIN5
4BE1 2AE64E     Ø959Ø  PASS2   LD      HL,(LINECT)
4BE4 7C         Ø96ØØ          LD      A,H
4BE5 B5         Ø961Ø          OR      L
4BE6 CA684C     Ø962Ø          JP      Z,ERROR2
4BE9 2AA44Ø     Ø963Ø          LD      HL,(4ØA4H)
4BEC 22F24E     Ø964Ø  PASS22  LD      (NXTLPT),HL
4BEF AF         Ø965Ø          XOR     A
```

Listing 3 continues

Listing 3 continued

```
4BF0 32E84E              LD    (ONFLAG),A
4BF3 110300              LD    DE,3
4BF6 19                  ADD   HL,DE
4BF7 23          GETBYT  INC   HL
4BF8 7E                  LD    A,(HL)
4BF9 B7                  OR    A
4BFA 200C                JR    NZ,COLON1
4BFC 23                  INC   HL
4BFD ED5BF84E            LD    DE,(PEND)
4C01 CD994C              CALL  COMPAR
4C04 20E6                JR    NZ,PASS22
4C06 1848                JR    FIN
4C08 FE3A        COLON1  CP    58
4C0A 2006                JR    NZ,COMMA
4C0C AF                  XOR   A
4C0D 32E84E              LD    (ONFLAG),A
4C10 18E5                JR    GETBYT
4C12 FE2C        COMMA   CP    44
4C14 200A                JR    NZ,CHEKON
4C16 3AE84E              LD    A,(ONFLAG)
4C19 B7                  OR    A
4C1A 28DB                JR    Z,GETBYT
4C1C 3E8D                LD    A,141
4C1E 1820                JR    GOFORT
4C20 FEA1        CHEKON  CP    161
4C22 2006                JR    NZ,KEYWRD
4C24 2F                  CPL
4C25 32E84E              LD    (ONFLAG),A
4C28 18CD                JR    GETBYT
4C2A FE8D        KEYWRD  CP    141
4C2C 2812                JR    Z,GOFORT
4C2E FE91                CP    145
4C30 280E                JR    Z,GOFORT
4C32 FECA                CP    202
4C34 280A                JR    Z,GOFORT
4C36 FE95                CP    149
4C38 20BD                JR    NZ,GETBYT
4C3A 08                  EX    AF,AF'
4C3B AF                  XOR   A
4C3C 32E84E              LD    (ONFLAG),A
4C3F 08                  EX    AF,AF'
4C40 CD0B4D      GOFORT  CALL  LOOKUP
4C43 B7                  OR    A
4C44 CC7D4C              CALL  Z,ERROR3
4C47 CB4F                BIT   1,A
4C49 20AD                JR    NZ,GETBYT+1
4C4B CDB44D              CALL  INSERT
4C4E 18A8                JR    GETBYT+1
4C50 2AF84E      FIN     LD    HL,(PEND)
4C53 23                  INC   HL
4C54 23                  INC   HL
4C55 22F940              LD    (40F9H),HL
4C58 21FA4E              LD    HL,FINI
4C5B CDA728              CALL  28A7H
4C5E C36E4C              JP    BASIC1
4C61 21044F      ERROR1  LD    HL,EMSG1
4C64 CDA728              CALL  28A7H
4C67 C9                  RET
4C68 21164F      ERROR2  LD    HL,EMSG2
4C6B CDA728              CALL  28A7H
4C6E CD641B      BASIC1  CALL  1B64H
4C71 AF                  XOR   A
4C72 329A40              LD    (409AH),A
4C75 2AF940              LD    HL,(40F9H)
4C78 2B                  DEC   HL
4C79 2B                  DEC   HL
4C7A C31E1D              JP    1D1EH
4C7D E5          ERROR3  PUSH  HL
4C7E 21294F              LD    HL,EMSG3
4C81 CDA728              CALL  28A7H
4C84 CD8C4C              CALL  ERRET
4C87 21F84B              LD    HL,GETBYT+1
4C8A E3                  EX    (SP),HL
4C8B C9                  RET
4C8C 2AF24E      ERRET   LD    HL,(NXTLPT)
4C8F 23                  INC   HL
4C90 23                  INC   HL
4C91 5E                  LD    E,(HL)
4C92 23                  INC   HL
4C93 56                  LD    D,(HL)
4C94 EB                  EX    DE,HL
4C95 CDA70F              CALL  0FA7H
4C98 C9                  RET
4C99 C5          COMPAR  PUSH  BC
4C9A D5                  PUSH  DE
4C9B E5                  PUSH  HL
4C9C F5                  PUSH  AF
4C9D 45                  LD    B,L
4C9E 4B                  LD    C,E
4C9F 6C                  LD    L,H
4CA0 5A                  LD    E,D
4CA1 AF                  XOR   A
4CA2 67                  LD    H,A
4CA3 57                  LD    D,A
4CA4 ED52                SBC   HL,DE
4CA6 2805                JR    Z,MSBEQU
4CA8 F2C04C              JP    P,HLGRTR
4CAB 180C                JR    HLLESS
4CAD 68          MSBEQU  LD    L,B
4CAE 59                  LD    E,C
4CAF AF                  XOR   A
4CB0 67                  LD    H,A
4CB1 57                  LD    D,A
4CB2 ED52                SBC   HL,DE
4CB4 2810                JR    Z,EQUAL
4CB6 F2C04C              JP    P,HLGRTR
4CB9 F1          HLLESS  POP   AF
4CBA 47                  LD    B,A
4CBB F6FF                OR    0FFH
4CBD 37                  SCF
```

Listing 3 continues

See List of Advertisers on Page 563

Listing 3 continued

```
4CBE 1809       10760        JR      EXITPT
4CC0 F1         10770 HLGRTR POP     AF
4CC1 47         10780        LD      B,A
4CC2 F6FF       10790        OR      0FFH
4CC4 1803       10800        JR      EXITPT
4CC6 F1         10810 EQUAL  POP     AF
4CC7 47         10820        LD      B,A
4CC8 AF         10830        XOR     A
4CC9 78         10840 EXITPT LD      A,B
4CCA E1         10850        POP     HL
4CCB D1         10860        POP     DE
4CCC C1         10870        POP     BC
4CCD C9         10880        RET
4CCE CD994C     10890 HLDESB CALL    COMPAR
4CD1 D8         10900        RET     C
4CD2 C5         10910        PUSH    BC
4CD3 4F         10920        LD      C,A
4CD4 2826       10930        JR      Z,ZEROIT
4CD6 0600       10940        LD      B,0
4CD8 CB7C       10950        BIT     7,H
4CDA 280F       10960        JR      Z,CLEAR
4CDC D5         10970        PUSH    DE
4CDD B7         10980        OR      A
4CDE CB1C       10990        RR      H
4CE0 CB1D       11000        RR      L
4CE2 F5         11010        PUSH    AF
4CE3 B7         11020        OR      A
4CE4 CB1A       11030        RR      D
4CE6 CB1B       11040        RR      E
4CE8 3816       11050        JR      C,CHKCRY
4CEA 04         11060        INC     B
4CEB B7         11070 CLEAR  OR      A
4CEC ED52       11080        SBC     HL,DE
4CEE CB40       11090        BIT     0,B
4CF0 2806       11100        JR      Z,SKIPIT
4CF2 F1         11110        POP     AF
4CF3 CB15       11120        RL      L
4CF5 CB14       11130        RL      H
4CF7 D1         11140        POP     DE
4CF8 B7         11150 SKIPIT OR      A
4CF9 79         11160        LD      A,C
4CFA C1         11170        POP     BC
4CFB C9         11180        RET
4CFC ED62       11190 ZEROIT SBC     HL,HL
4CFE 18F8       11200        JR      SKIPIT
4D00 F1         11210 CHKCRY POP     AF
4D01 3003       11220        JR      NC,CHKCR1
4D03 B7         11230        OR      A
4D04 1802       11240        JR      CHKCR1+2
4D06 13         11250 CHKCR1 INC     DE
4D07 37         11260        SCF
4D08 F5         11270        PUSH    AF
4D09 18DF       11280        JR      CLEAR-1
4D0B 23         11290 LOOKUP INC     HL
4D0C E5         11300        PUSH    HL
4D0D F5         11310        PUSH    AF
4D0E CD5A1E     11320        CALL    1E5AH
4D11 7A         11330        LD      A,D
4D12 B3         11340        OR      E
4D13 CA9F4D     11350        JP      Z,POSERR
4D16 F1         11360        POP     AF
4D17 2AEA4E     11370        LD      HL,(BLINE)
4D1A CD994C     11380        CALL    COMPAR
4D1D 2810       11390        JR      Z,FINDIT
4D1F 300A       11400        JR      NC,RETURN
4D21 2AEC4E     11410        LD      HL,(ELINE)
4D24 CD994C     11420        CALL    COMPAR
4D27 2806       11430        JR      Z,FINDIT
```

Listing 3 continues

References for Machine-Language Utility Programs

"Memories Are Made of This," Robert D. Randall, *80 Micro*, May 81, p. 146.
"The Flexible Scroller," Jeff Myers, *80 Micro*, May 81, p. 204.
"Shift Lock," Martin C. Hambel, *80 Micro*, May 81, p. 260.
"Comprs," Stephen Barker, *80 Micro*, May 81, p. 270.
"*Never Ready," Ron Balewski, *80 Micro*, July 81, p. 199.
"High-speed Data Tapes," Jim Glosser, *80 Micro*, July 81, p. 280.
"Datagen," Dan and Cass Lewart, *80 Micro*, Aug. 81, p. 168.
"Be a Super USR," Roger C. Alford, *80 Micro*, Aug. 81, p. 254.
"Basic Shorthand," Jared Radin, *80 Micro*, Aug. 81, p. 282.
"Three's Not a Crowd," David W. Mawdsley, *80 Micro*, Aug. 81, p. 292.
"*The Memory Expander," Tim Kenealy, *80 Micro*, Sept. 81, p. 174.
"Base Conversions," James Yowell Yelvington, *80 Micro*, Sept. 81, p. 186.
"High-speed Sorts," Richard R. Robson, *80 Micro*, Sept. 81, p. 220.
"Copyit," Ron Balewski, *80 Micro*, Oct. 81, p. 370.
"*Full Error," Harry and Ken Keairns, *80 Micro*, Oct. 81, p. 340.
"Merge for Level II," William J. Dalesandry Jr., *80 Micro*, Nov. 81, p. 284.
"*Customized Commands," Dale W. Rupert, *80 Micro*, Nov. 81, p. 292.
"*Basic—Enhanced Again," Mark Goodwin, *80 Micro*, Nov. 81, p. 384.
"The Spooler," Roger B. Gault, *80 Micro*, Dec. 81, p. 222.
"The Sentry," Jim Rastin, *80 Micro*, Dec. 81, p. 242.
"USR Usery," David H. Freese, Jr., *80 Micro*, Dec. 81, p. 248.
"The Freebie," John C. Adams, Jr., *80 Micro*, Dec. 81, p. 304.
"The Sargon Saver—Part II," Thomas L. Quindry, *80 Micro*, Dec. 81, p. 348.
"Array I/O," Norman Neff, *80 Micro*, Jan. 82, p. 200.
"*Lost in Basic," Mark C. Paxton, *80 Micro*, Jan. 82, p. 304.
"Tab Extender," David C. Hedinger, *80 Micro*, Feb. 82, p. 248.
"Error Code Expanded," Roger C. Alford, *80 Micro*, Feb. 82., p. 260.
"As the Screen Scrolls," M. Keller, *80 Micro*, Feb.82, p. 264.
"Varispeed," Bruce Evans, *80 Micro*, Mar. 82, p. 336.
"Bam," Jeff Byrkit, *80 Micro*, April 82, p. 167.
"New Restored," Ken Fordham, *80 Micro*, Jan. 80, p. 84.
"Spool and Despool," H.S. Gentry, *80 Micro*, Mar. 80, p. 46.
"Screenprint," Louis Frankenberg, *80 Micro*, May 80, p. 152; *80 Micro*, July 80, p. 10; *80 Micro*, Aug. 80, p. 16; *80 Micro*, Sept. 80, p. 14.
"Seespot (80 applications)," Dennis Kitsz, *80 Micro*, July 80, p. 26.
"*White," Jake Commander, *80 Micro*, July 80, p. 94.
"*Single-step Basic (80 Applications)," Dennis Kitsz, *80 Micro*, Aug. 80, p. 28.
"Free Space," David Cornell, *80 Micro*, Sept. 80, p. 68.
"*Unikey," Rowland Archer, Jr., *80 Micro*, Sept. 80, p. 76.
"Slow Scroll," Peter A. Lewis, *80 Micro*, Sept. 80, p. 202.
"*Get Serious," Roger L. Pape, *80 Micro*, Oct. 80, p. 93.
"Variable Scroll," William L. Colsher, *80 Micro*, Oct. 80, p. 134.
"Resurrect It," Thomas L. Quindry, *80 Micro*, Nov. 80, p. 206.
"Basic Line Compressor (Assembly Line)," William Barden, Jr., *80 Micro*, Jan. 81, p. 23.
"Zbug . . . Super Debug Monitor," Lt. John B. Harrell, *80 Micro*, Jan. 81, p. 131.
"*Auto Edit," Dan Rollins, *80 Micro*, Feb. 81, p. 144.
"Find It Fast," James Yowell Yelvington, *80 Micro*, Mar. 81, p. 180.
"Babydub," Dennis Kitsz, *80 Micro*, Mar. 81, p. 210; *80 Micro*, July 81, p. 16.
"*Constant Alternatives," Evan C. Hand, Sr., *80 Micro*, Mar. 81, p. 225.
"*Speedy Renumberer," Robert J. Dowd, *80 Micro*, Mar. 81, p. 256.
"*Bubble Sort (Assembly Line)," William Barden, Jr., *80 Micro*, April 81, p. 57.
"Block That Cursor," Ron Balewski, *80 Micro*, April 81, p. 297; *80 Micro*, July 81, p. 20.
"Smart Answers," James F. Williams, *80 Micro*, April 81, p. 298. *80 Micro*, Mar. 82, p. 24.

"Command Interpreter," Roger C. Alford, *80 Micro*, April 82, p. 244.
"CP80," Brian Cameron, *80 Micro*, April 82, p. 306; *80 Micro*, June/July 82, p. 28.
"Inverter (Soft Bits)," Roger Fuller, *80 Micro*, p. 34, 36.
"Horizontal Scroll (Soft Bits)," Roger Fuller, *80 Micro*, June/July 82, p. 40.
"For the Novice—Part I," Jay Chidsey, *80 Micro*, June/July 82, p. 148.
"Stepwise Refinement," B. Boasso, *80 Micro*, June/July 82, p. 148.
"Horizontal Scrolling," G. M. Foley, *80 Micro*, June/July 82, p. 318.

*Signifies article was used for NODOS 80 source material.

DOES STRING COMPRESSION HAVE YOU TIED UP IN KNOTS?

LET TRASHMAN CLEAN UP THE MESS!

TRASHMAN is a machine language utility for the TRS-80 Models I and III. It was written by Glenn Tesler, the author of FASTER, and can reduce BASIC's string compression time by 95% (see table below).

WHAT'S STRING COMPRESSION?

When a BASIC program changes a string (words, names, descriptions), it moves it to a new place in memory, and leaves a hole in the old place. Eventually, all available memory gets used up and BASIC has to push the strings together to free up some space. This takes time. Lots of time. The computer stops running for seconds or minutes, and you may even think it's "crashed". The keyboard won't work, and until all the strings have been collected, you just have to sit and wait. Then things run for a while, until string compression is needed again. And again.

If you're using your computer for business, that wastes your money. If you're using it personally, it wastes your time.

WHAT'S THE SOLUTION?

As soon as you start using TRASHMAN, those delays almost disappear. It uses less than 600 bytes of memory, plus 2 bytes for each active string. It works with other machine language programs and with all major operating systems. It's easy to use, comes with complete instructions, and can be copied to your own disks.

WHAT'S THE CATCH?

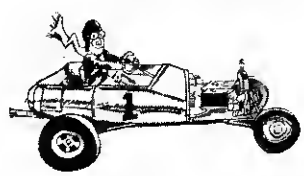
If a BASIC program uses only a few strings, very little time is wasted in string compression, and TRASHMAN won't be helpful. But, if hundreds of strings, including large string arrays, are used, TRASHMAN is just what you need.

# STRINGS	SECONDS DELAY NORMAL	SECONDS DELAY TRASHMAN	PERCENT IMPROVEMENT
250	11.8	0.7	94
500	45.8	1.6	96.5
1000	179.6	3.5	98
2000	713.2	7.8	98.9

Ask your software dealer for TRASHMAN, or order directly on our toll-free number. The price is just $39.95 (plus sales tax in California).

(All timings done on TRS-80 Model I. Model III 15% faster, but pct. improvements identical. Listing of timing program available on request.)

AMAZING PROGRAM SPEEDS UP BASIC



Your time is valuable, so why waste it on slow-running BASIC programs? PROSOFT's "FASTER" will analyze those programs *while they run*, then show you a simple change (usually one new line) that can reduce run-times by up to 50%.

Accounting systems, financial models, engineering and scientific programs all run faster; so do games. Large, complex programs improve the most, and "FASTER" is easy to use.

THIS ISN'T A COMPILER! Your BASIC programs remain readable and can be changed later on. While your programs run, "FASTER" counts how often each "variable" is used, then shows you the correct sequence for these variables. Afterwards, the computer finds them sooner, so your programs run faster.

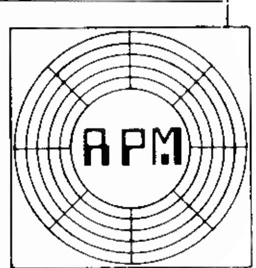
Does it really work? Yes! *Personal Computing* said so in their May, 1981 issue (p. 116); we've received many letters from customers who've gotten 20-50% improvements; and we will make you this offer:

Order "FASTER" now. Try it on your bread-and-butter programs. If you don't get an overall run-time reduction of at least 20%, return it within 30 days for a prompt and cheerful refund.

"FASTER runs on TRS-80 Models I and III, 16–48K, tape or disk

$29.95

ERRATIC DISK DRIVES



RPM measures the rotational speed and variation of your disk drives, and reveals a common cause of unexplained errors. Simple one-key operation, runs under any DOS, interchangeable between Models I and III. Shows current and average speeds, plus fluctuation history. Recovers from severe errors. Documentation explains how to adjust drives. Use RPM monthly for best results. 32-48K Model I or III disk: **$24.95**

QUICK COMPRESS

Small (276 bytes), fast (processes 800 lines in under 3 seconds) utility removes blanks and remarks from your BASIC programs. Produces smaller, faster programs, and doesn't alter the original logic.

16-48K Model I or III, tape or disk. **$19.95**

SPECIAL FASTER + QUICK Compress: **$39.95**

TO ORDER, CALL NOW, TOLL FREE

(800) 824-7888, Operator 422
CALIF: (800) 852-7777, Oper. 422
ALASKA/HAWAII: (800) 824-7919
FOR TECHNICAL INFORMATION CALL:
(213) 764-3131, or write to us.

PROSOFT ™

Dept. G Box 560 ✓91
North Hollywood, CA 91603

TERMS: We accept VISA, Mastercard, checks, Money Orders, C.O.D. and even cash. We pay shipping via surface UPS inside U.S.A. Please add $3.00 for Blue Label, 6% tax in California, and 15% outside North America (air shipment).

Listing 3 continued

```
4D29 3004       11440        JR    NC,FINDIT
4D2B E1         11450 RETURN POP   HL
4D2C C3AB4D     11460        JP    NOERR
4D2F ED4BE64E   11470 FINDIT LD    BC,(LINECT)
4D33 DD2AF34A   11480        LD    IX,(ARRAY)
4D37 DD6600     11490 LOOKLP LD    H,(IX+0)
4D3A DD2B       11500        DEC   IX
4D3C DD6E00     11510        LD    L,(IX+0)
4D3F DD2B       11520        DEC   IX
4D41 CD994C     11530        CALL  COMPAR
4D44 280C       11540        JR    Z,GOTMAT
4D46 DD2B       11550        DEC   IX
4D48 DD2B       11560        DEC   IX
4D4A 0B         11570        DEC   BC
4D4B 78         11580        LD    A,B
4D4C B1         11590        OR    C
4D4D 20E8       11600        JR    NZ,LOOKLP
4D4F C3AE4D     11610        JP    ERROR5
4D52 DD6600     11620 GOTMAT LD    H,(IX+0)
4D55 DD2B       11630        DEC   IX
4D57 DD6E00     11640        LD    L,(IX+0)
4D5A AF         11650        XOR   A
4D5B 32E94E     11660        LD    (GOFLAG),A
4D5E FD21774F   11670        LD    IY,ASCIIN
4D62 DD216D4F   11680        LD    IX,POWERS
4D66 0605       11690        LD    B,5
4D68 DD5E00     11700 LOOKL2 LD    E,(IX+0)
4D6B DD5601     11710        LD    D,(IX+1)
4D6E 3E00       11720        LD    A,0
4D70 0E30       11730        LD    C,'0'
4D72 CDCE4C     11740 LOOKL3 CALL  HLDESB
4D75 3021       11750        JR    NC,INCRA
4D77 B7         11760        OR    A
4D78 2811       11770        JR    Z,BLNKIT
4D7A 81         11780        ADD   A,C
4D7B 32E94E     11790        LD    (GOFLAG),A
4D7E FD7700     11800 LOOKL4 LD    (IY+0),A
4D81 FD23       11810        INC   IY
4D83 DD23       11820        INC   IX
4D85 DD23       11830        INC   IX
4D87 10DF       11840        DJNZ  LOOKL2
4D89 1810       11850        JR    ALLDON
4D8B 3AE94E     11860 BLNKIT LD    A,(GOFLAG)
4D8E B7         11870        OR    A
4D8F 2004       11880        JR    NZ,BLNK1
4D91 3E20       11890        LD    A,' '
4D93 18E9       11900        JR    LOOKL4
4D95 79         11910 BLNK1  LD    A,C
4D96 18E6       11920        JR    LOOKL4
4D98 3C         11930 INCRA  INC   A
4D99 18D7       11940        JR    LOOKL3
4D9B 3E01       11950 ALLDON LD    A,1
4D9D E1         11960        POP   HL
4D9E C9         11970        RET
4D9F F1         11980 POSERR POP   AF
4DA0 E1         11990        POP   HL
4DA1 FECA       12000        CP    202
4DA3 2806       12010        JR    Z,NOERR
4DA5 FE95       12020        CP    149
4DA7 2802       12030        JR    Z,NOERR
4DA9 AF         12040        XOR   A
4DAA C9         12050        RET
4DAB 3E02       12060 NOERR  LD    A,2
4DAD C9         12070        RET
4DAE 215B4F     12080 ERROR5 LD    HL,EMSG5
4DB1 C3814C     12090        JP    ERROR3+4
4DB4 7E         12100 INSERT LD    A,(HL)
4DB5 FE20       12110        CP    ' '
4DB7 2003       12120        JR    NZ,INSRT1
4DB9 23         12130        INC   HL
4DBA 18F8       12140        JR    INSERT
4DBC 0605       12150 INSRT1 LD    B,5
4DBE DD21774F   12160        LD    IX,ASCIIN
4DC2 DD7E00     12170 INSRT2 LD    A,(IX+0)
4DC5 FE20       12180        CP    ' '
4DC7 2004       12190        JR    NZ,INSRT3
4DC9 DD23       12200        INC   IX
4DCB 10F5       12210        DJNZ  INSRT2
4DCD F5         12220 INSRT3 PUSH  AF
4DCE 7E         12230        LD    A,(HL)
4DCF FE30       12240        CP    '0'
4DD1 FAEF4D     12250        JP    M,INSRT5
4DD4 FE3A       12260        CP    ':'
4DD6 F2EF4D     12270        JP    P,INSRT5
4DD9 F1         12280 INSRT6 POP   AF
4DDA 77         12290        LD    (HL),A
4DDB DD23       12300        INC   IX
4DDD 23         12310        INC   HL
4DDE DD7E00     12320        LD    A,(IX+0)
4DE1 10EA       12330        DJNZ  INSRT3
4DE3 7E         12340 INSRT4 LD    A,(HL)
4DE4 FE30       12350        CP    '0'
4DE6 F8         12360        RET   M
4DE7 FE3A       12370        CP    ':'
4DE9 F0         12380        RET   P
4DEA CD124E     12390        CALL  COMPRS
4DED 18F4       12400        JR    INSRT4
4DEF CDF44D     12410 INSRT5 CALL  EXPAND
4DF2 18E5       12420        JR    INSRT6
4DF4 E5         12430 EXPAND PUSH  HL
4DF5 08         12440        EX    AF,AF'
4DF6 D9         12450        EXX
4DF7 D1         12460        POP   DE
4DF8 2AF84E     12470        LD    HL,(PEND)
4DFB 23         12480        INC   HL
4DFC E5         12490        PUSH  HL
4DFD 22F84E     12500        LD    (PEND),HL
4E00 23         12510        INC   HL
4E01 E5         12520        PUSH  HL
4E02 CDCE4C     12530        CALL  HLDESB
4E05 E5         12540        PUSH  HL
4E06 C1         12550        POP   BC
4E07 D1         12560        POP   DE
4E08 E1         12570        POP   HL
4E09 EDB8       12580        LDDR
4E0B AF         12590        XOR   A
4E0C CD314E     12600        CALL  FIXPTS
4E0F 08         12610        EX    AF,AF'
4E10 D9         12620        EXX
4E11 C9         12630        RET
4E12 E5         12640 COMPRS PUSH  HL
4E13 08         12650        EX    AF,AF'
4E14 D9         12660        EXX
4E15 D1         12670        POP   DE
4E16 2AF84E     12680        LD    HL,(PEND)
4E19 2B         12690        DEC   HL
4E1A 22F84E     12700        LD    (PEND),HL
4E1D 23         12710        INC   HL
4E1E 23         12720        INC   HL
4E1F CDCE4C     12730        CALL  HLDESB
4E22 E5         12740        PUSH  HL
4E23 C1         12750        POP   BC
4E24 D5         12760        PUSH  DE
4E25 E1         12770        POP   HL
4E26 23         12780        INC   HL
4E27 EDB0       12790        LDIR
4E29 AF         12800        XOR   A
4E2A 2F         12810        CPL
4E2B CD314E     12820        CALL  FIXPTS
4E2E 08         12830        EX    AF,AF'
4E2F D9         12840        EXX
4E30 C9         12850        RET
4E31 2AF24E     12860 FIXPTS LD    HL,(NXTLPT)
4E34 5E         12870 FIXPT1 LD    E,(HL)
4E35 23         12880        INC   HL
4E36 56         12890        LD    D,(HL)
4E37 B7         12900        OR    A
4E38 2813       12910        JR    Z,ADDON
4E3A 1B         12920        DEC   DE
4E3B 72         12930 FIXPT2 LD    (HL),D
4E3C 2B         12940        DEC   HL
4E3D 73         12950        LD    (HL),E
```

Listing 3 continues

Listing 3 continued

```
4E3E D5         12960          PUSH   DE
4E3F C1         12970          POP    BC
4E40 EB         12980          EX     DE,HL
4E41 ED5BF84E   12990          LD     DE,(PEND)
4E45 CD994C     13000          CALL   COMPAR
4E48 D0         13010          RET    NC
4E49 C5         13020          PUSH   BC
4E4A E1         13030          POP    HL
4E4B 18E7       13040          JR     FIXPT1
4E4D 13         13050 ADDON    INC    DE
4E4E 18EB       13060          JR     FIXPT2
4E50 DD21924E   13070 OPTION   LD     IX,AMSG1
4E54 FD21EA4E   13080          LD     IY,BLINE
4E58 0604       13090          LD     B,4
4E5A C5         13100 LOOPOP   PUSH   BC
4E5B DD6E00     13110          LD     L,(IX+0)
4E5E DD6601     13120          LD     H,(IX+1)
4E61 CDA728     13130          CALL   28A7H
4E64 2AA740     13140          LD     HL,(40A7H)
4E67 2B         13150          DEC    HL
4E68 2B         13160          DEC    HL
4E69 CDD905     13170          CALL   05D9H
4E6C AF         13180          XOR    A
4E6D B0         13190          OR     B
4E6E 280D       13200          JR     Z,NEXTOP
4E70 CD5A1E     13210          CALL   1E5AH
4E73 7A         13220          LD     A,D
4E74 B3         13230          OR     E
4E75 2812       13240          JR     Z,ERROR4
4E77 FD7300     13250          LD     (IY+0),E
4E7A FD7201     13260          LD     (IY+1),D
4E7D DD23       13270 NEXTOP   INC    IX
4E7F DD23       13280          INC    IX
4E81 FD23       13290          INC    IY
4E83 FD23       13300          INC    IY
4E85 C1         13310          POP    BC
4E86 10D2       13320          DJNZ   LOOPOP
4E88 C9         13330          RET
4E89 21444F     13340 ERROR4   LD     HL,EMSG4
4E8C CDA728     13350          CALL   28A7H
4E8F C1         13360          POP    BC
4E90 18C8       13370          JR     LOOPOP
4E92 9A4E       13380 AMSG1    DEFW   MESG1
4E94 AD4E       13390          DEFW   MESG2
4E96 BD4E       13400          DEFW   MESG3
4E98 D34E       13410          DEFW   MESG4
4E9A 0D         13420 MESG1    DEFB   0DH
4E9B 45         13430          DEFM   'Enter start line '
4EAC 00         13440          DEFB   0
4EAD 45         13450 MESG2    DEFM   'Enter end line '
4EBC 00         13460          DEFB   0
4EBD 45         13470 MESG3    DEFM   'Enter new start line '
4ED2 00         13480          DEFB   0
4ED3 45         13490 MESG4    DEFM   'Enter increment '
4EE3 00         13500          DEFB   0
4EE4 0000       13510 INDEX    DEFW   0
4EE6 0000       13520 LINECT   DEFW   0
4EE8 00         13530 ONFLAG   DEFB   0
4EE9 00         13540 COFLAG   DEFB   0
4EEA 0000       13550 BLINE    DEFW   0
4EEC 0000       13560 ELINE    DEFW   0
4EEE 0000       13570 SLINE    DEFW   0
4EF0 0000       13580 INCR3    DEFW   0
4EF2 0000       13590 NXTLPT   DEFW   0
4EF4 0000       13600 CURLIN   DEFW   0
4EF6 0000       13610 CURLPT   DEFW   0
4EF8 0000       13620 PEND     DEFW   0
4EFA 0D         13630 FINI     DEFB   0DH
4EFB 46         13640          DEFM   'Finished'
4F03 00         13650          DEFB   0
4F04 0D         13660 EMSG1    DEFB   0DH
4F05 4C         13670          DEFM   'Line # too large'
4F15 00         13680          DEFB   0
4F16 0D         13690 EMSG2    DEFB   0DH
4F17 52         13700          DEFM   'Range value error'
4F28 00         13710          DEFB   0
4F29 0D         13720 EMSG3    DEFB   0DH
4F2A 47         13730          DEFM   'GOTO/GOSUB missing line #'
4F43 00         13740          DEFB   0
4F44 50         13750 EMSG4    DEFM   'Parameter not numeric'
4F59 0D         13760          DEFB   0DH
4F5A 00         13770          DEFB   0
4F5B 0D         13780 EMSG5    DEFB   0DH
4F5C 55         13790          DEFM   'Undefined line #'
4F6C 00         13800          DEFB   0
4F6D 1027       13810 POWERS   DEFW   10000
4F6F E803       13820          DEFW   1000
4F71 6400       13830          DEFW   100
4F73 0A00       13840          DEFW   10
4F75 0100       13850          DEFW   1
0005            13860 ASCIIN   DEFS   5
4F7C C37344     13870 PATCHC   JP     BASIC
4473            13880 BASIC    EQU    4473H
4895            13890 CHECK1   EQU    4895H
44DF            13900          ORG    44DFH     ;SET COMMAND ADDRESS
44DF F54A       13910          DEFW   START7
02B2            13920          END    02B2H
00000 TOTAL ERRORS
```

Program Listing 4

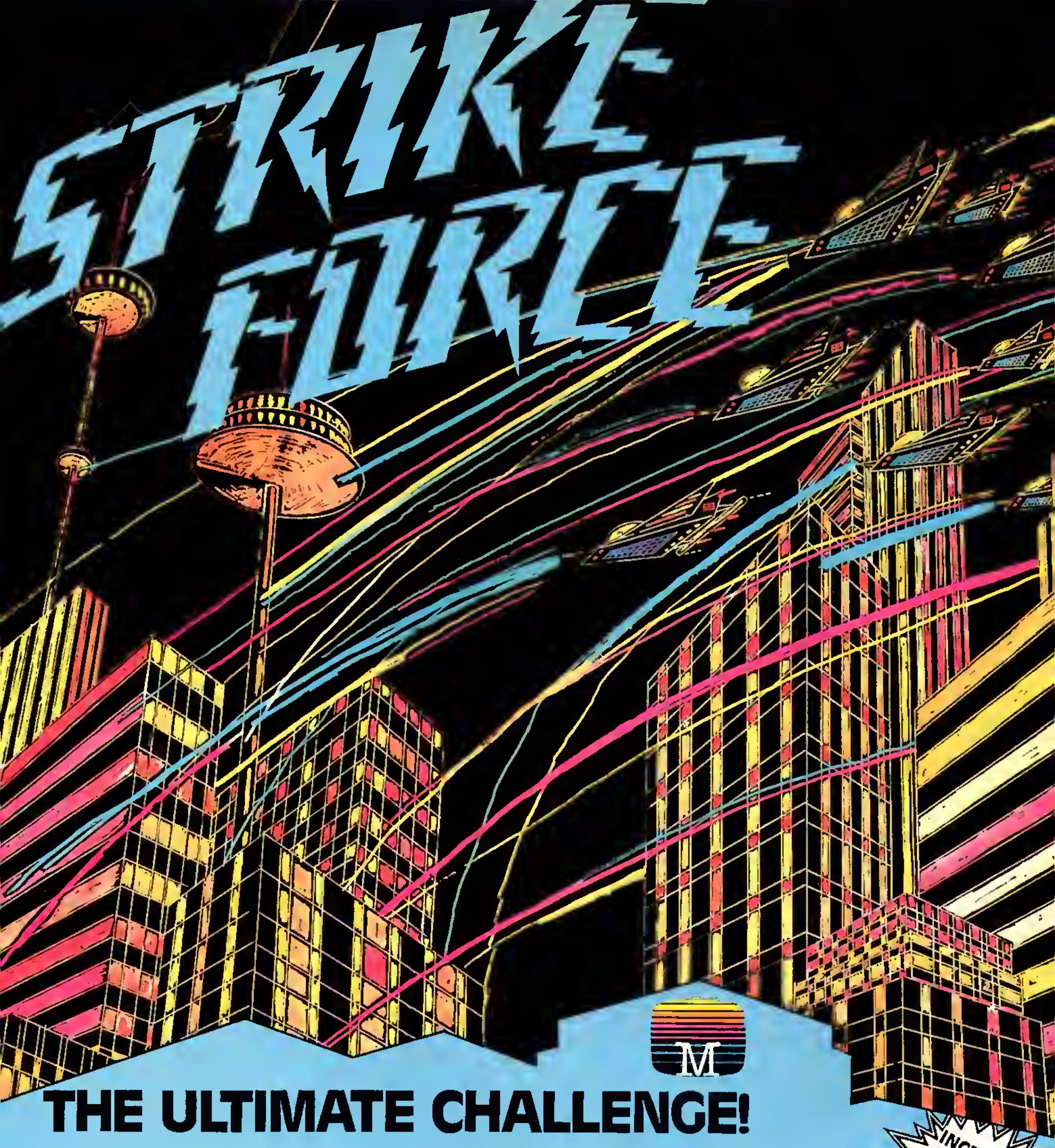
```
                13930 ;        NODOS80-D       FUNCTIONS
                13940 ;        DUAL, ROUTE, SINGLE KEY ENTRY, DEBOUNCE
                13950 ;        SORT, AUTO EDIT, SINGLE STEP BASIC
                13960 ;
                13970 ;        DUAL AND ROUTE
                13980 ;        THOMAS L. QUINDRY, NEW CODE
4F7C            13990          ORG    4F7CH
4F7C 2A1E40     14000 PATCHC   LD     HL,(401EH)      ;VIDEO DCB
4F7F 22B24F     14010          LD     (VIDEO+1),HL
4F82 2A2640     14020          LD     HL,(4026H)      ;PRINTER DCB
4F85 22D54F     14030          LD     (LPRINT),HL
4F88 C31D53     14040          JP     CLENUP
4F8B CD994F     14050 DUALON   CALL   DUALOF
4F8E CDA64F     14060          CALL   PRSTAT
4F91 C0         14070          RET    NZ
4F92 21AE4F     14080          LD     HL,DUAL
4F95 221E40     14090          LD     (401EH),HL
4F98 C9         14100          RET
4F99 2AB24F     14110 DUALOF   LD     HL,(VIDEO+1)
4F9C 221E40     14120          LD     (401EH),HL
4F9F 2AD54F     14130          LD     HL,(LPRINT)
4FA2 222640     14140          LD     (4026H),HL
4FA5 C9         14150          RET
4FA6 3AE837     14160 PRSTAT   LD     A,(37E8H)       ;PRINTER STATUS
4FA9 E6F0       14170          AND    0F0H
4FAB FE30       14180          CP     30H
4FAD C9         14190          RET
4FAE CDB44F     14200 DUAL     CALL   LPR             ;OUTPUT TO PRINTER AND VIDEO

4FB1 C30000     14210 VIDEO    JP     0000
4FB4 F5         14220 LPR      PUSH   AF              ;LINEPRINTER FILTER
4FB5 D5         14230          PUSH   DE
4FB6 79         14240          LD     A,C
4FB7 FE0E       14250          CP     14
4FB9 280D       14260          JR     Z,NOPRT
4FBB FE0F       14270          CP     15
4FBD 2809       14280          JR     Z,NOPRT
4FBF FE7F       14290          CP     127
4FC1 3802       14300          JR     C,PRT
4FC3 3E20       14310          LD     A,20H
4FC5 CD3B00     14320 PRT      CALL   003BH
4FC8 D1         14330 NOPRT    POP    DE
4FC9 F1         14340          POP    AF
4FCA C9         14350          RET
4FCB CD994F     14360 PRDO     CALL   DUALOF          ;ROUTE PRINTER TO DISPLAY
```

Listing 4 continues

Listing 4 continued

```
4FCE 21E54F   14370          LD     HL,VID
4FD1 222640   14380          LD     (4026H),HL
4FD4 C9       14390          RET
4FD5 0000     14400 LPRINT   DEFW   0000
4FD7 CD994F   14410 DOPR     CALL   DUALOF          ;ROUTE DISPLAY TO PRINTER
4FDA CDA64F   14420          CALL   PRSTAT
4FDD C0       14430          RET    NZ
4FDE 21B44F   14440          LD     HL,LPR
4FE1 221E40   14450          LD     (401EH),HL
4FE4 C9       14460          RET
4FE5 F5       14470 VID      PUSH   AF              ;VIDEO ROUTINE
4FE6 D5       14480          PUSH   DE
4FE7 79       14490          LD     A,C
4FE8 CD3300   14500          CALL   0033H
4FEB D1       14510          POP    DE
4FEC F1       14520          POP    AF
4FED C9       14530          RET
              14540 ;        SINGLE KEY COMMAND ENTRY ADAPTED FROM
              14550 ;        UNIKEY
              14560 ;        ROWLAND ARCHER JR, 80 MICRO, SEPT 80, P76
4FEE 2A1640   14570 PATCH8   LD     HL,(4016H)      ;KEYBOARD DCB
4FF1 E5       14580          PUSH   HL
4FF2 2A0C51   14590          LD     HL,(KEYDR1+1)
4FF5 221640   14600          LD     (4016H),HL
4FF8 E1       14610          POP    HL
4FF9 220C51   14620          LD     (KEYDR1+1),HL
4FFC 22AF51   14630          LD     (KEYDR2+1),HL
4FFF C37344   14640          JP     BASIC
5002 3ECD     14650 BOUNCE   LD     A,0CDH          ;GET DELAY
5004 320851   14660          LD     (DEBOU1),A
5007 32AB51   14670          LD     (DEBOU2),A
500A 211E50   14680          LD     HL,DELAY
500D 220951   14690          LD     (DEBOU1+1),HL
5010 22AC51   14700          LD     (DEBOU2+1),HL
5013 C9       14710          RET
5014 211801   14720 NOBOU    LD     HL,0118H                  ;NO DELAY
5017 220851   14730          LD     (DEBOU1),HL
501A 22AB51   14740          LD     (DEBOU2),HL
501D C9       14750          RET
501E C5       14760 DELAY    PUSH   BC
501F 010002   14770          LD     BC,0200H
5022 C1       14780          POP    BC
5023 C9       14790          RET
5024 00       14800 KEEPER   DEFB   0               ;SAVE KEYBOARD ENTRY
5025 52       14810 USTR     DEFM   'RUN'
5028 0D       14820          DEFB   0DH
5029 00       14830          DEFB   0
003B          14840 REST     DEFS   59
5065 00       14850          DEFB   0
5066 00       14860 OSFLAG   DEFB   0
5067 0000     14870 OSPTR    DEFW   0
5069 50       14880 LA       DEFM   'PRINT@'
506F 00       14890          DEFB   0
5070 45       14900 LB       DEFM   'ELSE'
5074 00       14910          DEFB   0
5075 43       14920 LC       DEFM   'CHR$('
507A 00       14930          DEFB   0
507B 44       14940 LD       DEFM   'DATA'
507F 00       14950          DEFB   0
5080 52       14960 LE       DEFM   'RIGHT$('
5087 00       14970          DEFB   0
5088 46       14980 LF       DEFM   'FOR'
508B 00       14990          DEFB   0
508C 47       15000 LG       DEFM   'GOTO'
5090 00       15010          DEFB   0
5091 52       15020 LH       DEFM   'RND('
5095 00       15030          DEFB   0
5096 49       15040 LI       DEFM   'INPUT"'
509C 00       15050          DEFB   0
509D 52       15060 LJ       DEFM   'READ'
50A1 00       15070          DEFB   0
50A2 49       15080 LK       DEFM   'INKEY$'
50A8 00       15090          DEFB   0
50A9 4C       15100 LL       DEFM   'LEN('
50AD 00       15110          DEFB   0
50AE 41       15120 LM       DEFM   'ASC('
50B2 00       15130          DEFB   0
50B3 4E       15140 LN       DEFM   'NEXT'
50B7 00       15150          DEFB   0
50B8 50       15160 LO       DEFM   'POKE'
50BC 00       15170          DEFB   0
50BD 50       15180 LP       DEFM   'PEEK('
50C2 00       15190          DEFB   0
50C3 4C       15200 LQ       DEFM   'LEFT$('
50C9 00       15210          DEFB   0
50CA 52       15220 LR       DEFM   'RETURN'
50D0 00       15230          DEFB   0
50D1 47       15240 LS1      DEFM   'GOSUB'
50D6 00       15250          DEFB   0
50D7 54       15260 LT       DEFM   'TAB('
50DB 00       15270          DEFB   0
50DC 55       15280 LU       DEFM   'USING'
50E1 00       15290          DEFB   0
50E2 53       15300 LV       DEFM   'STRING$('
50EA 00       15310          DEFB   0
50EB 4D       15320 LW       DEFM   'MID$('
50F0 00       15330          DEFB   0
50F1 53       15340 LX       DEFM   'SET('
50F5 00       15350          DEFB   0
50F6 54       15360 LY       DEFM   'THEN'
50FA 00       15370          DEFB   0
50FB 52       15380 LZ       DEFM   'RESET('
5101 00       15390          DEFB   0
5102 3A6650   15400 ONESTR   LD     A,(OSFLAG)
5105 B7       15410          OR     A
5106 2078     15420          JR     NZ,SUBST
5108 1801     15430 DEBOU1   JR     KEYDR1
510A 00       15440          NOP
510B CD0251   15450 KEYDR1   CALL   ONESTR
510E 322450   15460          LD     (KEEPER),A
5111 3A4038   15470          LD     A,(3840H)       ;CHECK FOR DOWN ARROW
5114 FE10     15480          CP     16
5116 3A2450   15490          LD     A,(KEEPER)
5119 2804     15500          JR     Z,ARROW
511B B7       15510          OR     A
511C C8       15520          RET    Z
511D 180B     15530          JR     ARROW1
511F 3A4038   15540 ARROW    LD     A,(3840H)
5122 B7       15550          OR     A
5123 20FA     15560          JR     NZ,ARROW        ;MUST RELEASE
5125 3E1A     15570          LD     A,1AH
5127 322450   15580          LD     (KEEPER),A
512A 3A8038   15590 ARROW1   LD     A,(3880H)
512D B7       15600          OR     A
512E 3A2450   15610          LD     A,(KEEPER)
5131 C8       15620          RET    Z
5132 3A4038   15630          LD     A,(3840H)
5135 FE02     15640          CP     2
5137 2858     15650          JR     Z,DEFINE
5139 3A2450   15660          LD     A,(KEEPER)
513C FE1A     15670          CP     1AH             ;DOWN ARROW
513E 2815     15680          JR     Z,SUBMOD
5140 3A2450   15690          LD     A,(KEEPER)
5143 CBAF     15700          RES    5,A
5145 FE29     15710          CP     41
5147 3A2450   15720          LD     A,(KEEPER)
514A D8       15730          RET    C
514B CBAF     15740          RES    5,A
514D FE5B     15750          CP     5BH
514F 3A2450   15760          LD     A,(KEEPER)
5152 3801     15770          JR     C,SUBMOD
5154 C9       15780          RET
5155 E5       15790 SUBMOD   PUSH   HL
5156 3A2450   15800          LD     A,(KEEPER)
5159 FE1A     15810          CP     1AH             ;DOWN ARROW
515B 2005     15820          JR     NZ,KEYWR
515D 212550   15830          LD     HL,USTR
5160 1815     15840          JR     SRCHDN
5162 216950   15850 KEYWR    LD     HL,LA
5165 3A2450   15860          LD     A,(KEEPER)
5168 CBAF     15870          RES    5,A
516A D641     15880          SUB    41H
```

Listing 4 continues


STRIKE-
FORCE-
M

Listing 4 continued

```
516C B7                OR      A
516D 2808              JR      Z,SRCHDN
516F 47                LD      B,A
5170 7E         NXTC   LD      A,(HL)
5171 23                INC     HL
5172 B7                OR      A
5173 20FB              JR      NZ,NXTC
5175 10F9              DJNZ    NXTC
5177 226750     SRCHDN LD      (OSPTR),HL
517A 3E01       SETMD  LD      A,1
517C 326650            LD      (OSFLAG),A
517F E1                POP     HL
5180 E5         SUBST  PUSH    HL
5181 2A6750            LD      HL,(OSPTR)
5184 7E                LD      A,(HL)
5185 B7                OR      A
5186 2003              JR      NZ,NOTEND
5188 326650            LD      (OSFLAG),A
518B 23         NOTEND INC     HL
518C 226750            LD      (OSPTR),HL
518F E1                POP     HL
5190 C9                RET
5191 3A8038     DEFINE LD      A,(3880H)        ;SHIFT KEY
5194 B7                OR      A
5195 3A2450            LD      A,(KEEPER)
5198 2002              JR      NZ,DEF
519A B7                OR      A
519B C9                RET
519C C5         DEF    PUSH    BC
519D E5                PUSH    HL
519E 21E451            LD      HL,STRTDF
51A1 CDDB51            CALL    PUTSTR
51A4 212550            LD      HL,USTR
51A7 0640              LD      B,64
51A9 E5         GETC   PUSH    HL
51AA C5                PUSH    BC
51AB 1801       DEBOU2 JR      KEYDR2
51AD 00                NOP
51AE CD0000     KEYDR2 CALL    0000H
51B1 C1                POP     BC
51B2 E1                POP     HL
51B3 B7                OR      A
51B4 28F3              JR      Z,GETC
51B6 322450            LD      (KEEPER),A
51B9 3A4038            LD      A,(3840H)
51BC FE02              CP      2
51BE 2006              JR      NZ,NTENDF
51C0 3A8038            LD      A,(3880H)
51C3 B7                OR      A
51C4 200A              JR      NZ,ENDDEF
51C6 3A2450     NTENDF LD      A,(KEEPER)
51C9 77                LD      (HL),A
51CA 23                INC     HL
51CB CD3A03            CALL    033AH
51CE 10D9              DJNZ    GETC
51D0 AF         ENDDEF XOR     A
51D1 77                LD      (HL),A
51D2 21F551            LD      HL,ENDMSG
51D5 CDDB51            CALL    PUTSTR
51D8 E1                POP     HL
51D9 C1                POP     BC
51DA C9                RET
51DB 7E         PUTSTR LD      A,(HL)
51DC B7                OR      A
51DD C8                RET     Z
51DE CD3A03            CALL    033AH
51E1 23                INC     HL
51E2 18F7              JR      PUTSTR
51E4 0D         STRTDF DEFB    0DH
51E5 44                DEFM    'Define String:'
51F3 0D                DEFB    0DH
51F4 00                DEFB    0
51F5 0D         ENDMSG DEFB    0DH
51F6 45                DEFM    'End Definition'
5204 0D                DEFB    0DH
5205 00                DEFB    0
                ;      SORT ARRAY STRING ADAPTED FROM
                ;      BUBBLE SORT, THE ASSEMBLY LINE
                ;      WILLIAM BARDEN JR, 80 MICRO, APR 81, P53
                ;      SORT ARRAY STARTING WITH ZERO ELEMENT
                ;      COMMAND IS OF THE FORM CMD"O",A$(0)
                ;      ARRAY MUST BE DIMENSIONED
5206 C1         SABUBL POP     BC
5207 F1                POP     AF
5208 D1                POP     DE
5209 E1                POP     HL
520A E5                PUSH    HL
520B D5                PUSH    DE
520C F5                PUSH    AF
520D D7                RST     16
520E D7                RST     16
520F FE22              CP      '"'              ;CHECK FORMAT
5211 2803              JR      Z,SAB01A
5213 C3A024     SABERR JP      24A0H
5216 D7         SAB01A RST     16
5217 FE2C              CP      ','
5219 20F8              JR      NZ,SABERR
521B E5                PUSH    HL
521C D7         STRING RST     16               ;MUST BE (0)
521D FE3A              CP      ':'
521F CA1352            JP      Z,SABERR
5222 B7                OR      A
5223 CA1352            JP      Z,SABERR
5226 FE24              CP      '$'
5228 20F2              JR      NZ,STRING
522A D7                RST     16
522B CF                RST     8
522C 28                DEFB    '('
522D CF                RST     8
522E 30                DEFB    '0'
522F CF                RST     8
```

Listing 4 continues

the Model III. Nodelay removes this function.

EDT is another toggle command, allowing an autoedit function. Using the shift, enter keys puts you in the edit mode for the line after that already in the current line pointer in RAM. The proper way to use the autoedit function is to specify EDT, then use the Edit command of Basic for the first line to be edited. Consecutive lines are automatically put in the edit mode with the shift, enter combination. Entering the EDT command a second time disables the autoedit function. As with Unikey, you can enable or disable the autoedit function as often as you like.

"The proper way to use the autoedit function is to specify EDT, then use the End command of Basic for the first line to be edited."

SNGL is a toggle command like Unikey and EDT. Entering the command allows you to single-step through Basic. You must enter the shift key to continue operation. A small delay occurs before you invoke the next Basic command. You can enable or disable this function as often as you like.

Dual or Dual Y routes anything output through the Print command to a printer as well as the video. This command isn't invoked if the printer is not ready. This avoids locking up computers without printers. Dual N disables the command. You can use these commands within a Basic program by taking certain precautions. The Pack utility removes the space between Dual Y and Dual N, and the command is not recognized. If the printer is busy when you invoke the Dual command, it won't recognize the command. When the Dual or Dual Y commands are not recognized, the command disables all route functions.

The PRDO command routes LPRINT commands to the video instead. This is especially useful when you don't have a printer or prefer the text to be printed to video such as when debugging a program. If you don't

Save more money (and data) with our hard disk drive
Introducing the new, low-priced Winchester hard disk drive from A.M. Electronics. Designed to put the power of hard disk data storage within reach of more system owners than ever before.

More megabytes per buck
Our drive features an unformatted data capacity of 6.7 megabytes, with an average access time of 75 milliseconds. Up to four drives can be daisy-chain connected for even greater capacity.

Ready to run with TRS-80
Each drive comes complete with controller board, ST-506 interface, and a DOS Plus 4.0i for complete plug compatibility with your TRS-80 system.

Quality you can count on
Manufactured from the finest components, the A.M. Electronics hard disk drive has an expected life rating of five years when used 50% of the time.

Power consumption is unusually low: typically one amp less than other Winchester systems.

The disk and read/write heads are fully enclosed in a module using an integral air recirculation system with an absolute filter. You can depend on stable operation and high reliability in any office environment.

Think of the possibilities
Word processing, large mailing lists, database management, financial planning and forecasting . . . demanding applications like these and many more are now possible—and affordable—with the new A.M. Electronics hard disk drive.

For more information, or to find the location of your nearest dealer, please contact: A.M. Electronics, Inc., 3446 Washtenaw Ave., Ann Arbor, MI 48104, 313/973-2075, Telex: 220821 HPC. In Europe, contact A.M. Electronics, Ltd., 10 Barley Mow, Barley Mow Passage, Chiswick, London W4 4PH, England, Tel. (01)994-6477, Telex: 8811418.

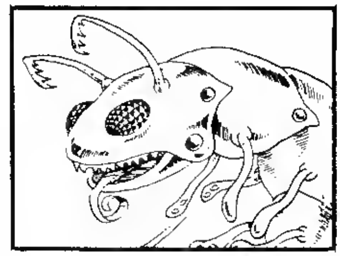
A.M. ELECTRONICS INC
THE POWER BEHIND THE DRIVES™

~38

See Us At
COMDEX
Nov. 28-Dec. 2, 1982

TM TRS-80 is a trademark of Tandy Corp.

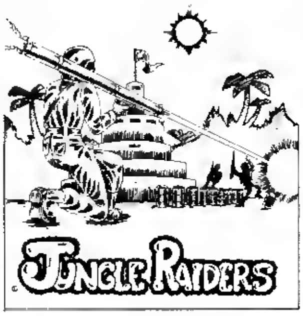
Extra! Extra!



Insect Frenzy · Page 5

Volume 4 | All The News That's Fit To *ZAP!*

Users vote no to the same OLD ARCADE GAMES



(DV 1982) Blurry eyed users have turned to **DISPLAYED VIDEO** to answer their need for new and exciting arcade games. In response, **DISPLAYED VIDEO** has announced eight new programs for the TRS-80* Models I-III. These arcade type games feature sound, graphics, joystick compatibility and are written in machine language for maximum speed! Both disk and tape versions allow the user to save high scores, a feature not usually found on cassette based games. Maze enthusiasts seem to like Ghost Hunter and Killer Beetles, while gun slingers look toward Insect Frenzy, Jungle Raiders, Space Shootout, Alien Cresta and Battle Stations for excitement. A Game that does not fall into these categories is Hoppy. It features wild drivers, sinking turtles, and hungry alligators. These programs are distributed exclusively by **DISPLAYED VIDEO** and written by Dubois and McNamara. Pricing for these programs is **$15.95** for tape and **$19.95** for disk. Reliable sources inside the company indicate Killer Gorilla will be available by the time you read this.

TO ORDER SEND CHECK OR MONEY ORDER OR CALL (517) 542-3280

Visit Our Two
RETAIL LOCATIONS:
111 Marshall St., Litchfield, MI 49252
(517) 542-3280
or
886 Ecorse Rd., Ypsilanti, MI 48197
(313) 426-5086/(313) 482-4424

*Authorized Radio Shack Dealership

DISPLAYED VIDEO

DEALER DISCOUNTS AVAILABLE

Joysticks for above programs **$39.95**
Add $2.00 for Shipping
Add $1.50 for COD
Mich. res. add 4 percent tax

*TRS 80 is a trademark of the Tandy Corp.
Prices subject to change without notice.

Listing 4 continued

```
5230 29                   DEFB    ')'
5231 E1                   POP     HL
5232 D7                   RST     16            ;POINT HL TO VARIABLE NAME
5233 CD0D26               CALL    260DH         ;VARPTR
5236 E5                   PUSH    HL
5237 D5                   PUSH    DE            ;STRING ADDRESS HERE
5238 DDE1                 POP     IX
523A DDE5        SAB010   PUSH    IX
523C DD4EFE               LD      C,(IX+254)    ;(IX-2)
523F DD46FF               LD      B,(IX+255)    ;(IX-1)
5242 0B                   DEC     BC
5243 FD210000             LD      IY,0
5247 C5          SAB020   PUSH    BC
5248 DD7E00               LD      A,(IX)
524B B7                   OR      A
524C 2857                 JR      Z,SAB070
524E DD7E03               LD      A,(IX+3)
5251 DD4600               LD      B,(IX)
5254 90                   SUB     B
5255 ED44                 NEG
5257 4F                   LD      C,A
5258 FA5E52               JP      M,SAB030
525B DD4603               LD      B,(IX+3)
525E DD6E01      SAB030   LD      L,(IX+1)
5261 DD6602               LD      H,(IX+2)
5264 DD5E04               LD      E,(IX+4)
5267 DD5605               LD      D,(IX+5)
526A E5                   PUSH    HL
526B D5                   PUSH    DE
526C DD7E03               LD      A,(IX+3)
526F B7                   OR      A
5270 2004                 JR      NZ,SAB040
5272 D1                   POP     DE
5273 E1                   POP     HL
5274 1813                 JR      SAB060
5276 1A          SAB040   LD      A,(DE)
5277 96                   SUB     (HL)
5278 ED44                 NEG
527A 2005                 JR      NZ,SAB050
527C 13                   INC     DE
527D 23                   INC     HL
527E 10F6                 DJNZ    SAB040
5280 79                   LD      A,C
5281 D1          SAB050   POP     DE
5282 E1                   POP     HL
5283 B7                   OR      A
5284 281F                 JR      Z,SAB070
5286 FAA552               JP      M,SAB070
5289 DD7504      SAB060   LD      (IX+4),L
528C DD7405               LD      (IX+5),H
528F DD7301               LD      (IX+1),E
5292 DD7202               LD      (IX+2),D
5295 DD7E00               LD      A,(IX)
5298 DD4603               LD      B,(IX+3)
529B DD7703               LD      (IX+3),A
529E DD7000               LD      (IX),B
52A1 FD214000             LD      IY,40H
52A5 C1          SAB070   POP     BC
52A6 DD23        SAB080   INC     IX
52A8 DD23                 INC     IX
52AA DD23                 INC     IX
52AC 0B                   DEC     BC
52AD 78                   LD      A,B
52AE B1                   OR      C
52AF 2096                 JR      NZ,SAB020
52B1 FDE5                 PUSH    IY
52B3 F1                   POP     AF
52B4 DDE1                 POP     IX
52B6 2002                 JR      NZ,SAB090
52B8 1880                 JR      SAB010
52BA E1          SAB090   POP     HL
52BB F1                   POP     AF
52BC D1                   POP     DE
52BD C1                   POP     BC
52BE C9                   RET
                 ;        AUTO EDIT
                 ;        DAN ROLLINS, 80 MICRO, FEB 81, P144
                 ;        <SHIFT><ENTER> TO ACTIVATE
52BF 3AAF41      PATCHA   LD      A,(41AFH)
52C2 F5                   PUSH    AF
52C3 2AB041               LD      HL,(41B0H)
52C6 E5                   PUSH    HL
52C7 3ADE52               LD      A,(STOREA)
52CA 32AF41               LD      (41AFH),A
52CD 2ADF52               LD      HL,(STOREA+1)
52D0 22B041               LD      (41B0H),HL
52D3 E1                   POP     HL
52D4 F1                   POP     AF
52D5 32DE52               LD      (STOREA),A
52D8 22DF52               LD      (STOREA+1),HL
52DB C37344               JP      BASIC
52DE C3E152      STOREA   JP      ATOEDT
52E1 3A8038      ATOEDT   LD      A,(3880H)     ;CHECK SHIFT KEY
52E4 B7                   OR      A
52E5 28F7                 JR      Z,STOREA
52E7 ED5BEC40             LD      DE,(40ECH)    ;CURRENT LINE POINTER
52EB 13                   INC     DE
52EC CD2C1B               CALL    1B2CH         ;GET ADDRESS FOR LINE NO.
52EF C5                   PUSH    BC
52F0 E1                   POP     HL
52F1 23                   INC     HL
52F2 23                   INC     HL
52F3 CD3F1B               CALL    1B3FH         ;LD     HL,(HL) ROM ROUTINE
52F6 C3662E               JP      2E66H         ;JUMP WITHIN EDIT
                 ;        SINGLE STEP, 80 APPLICATIONS
                 ;        DENNIS KITSZ, 80 MICRO, AUG 80, P28
                 ;        SINGLE STEP THROUGH A BASIC PROGRAM
52F9 2A0440      PATCHB   LD      HL,(4004H)
52FC E5                   PUSH    HL
52FD 2A1B53               LD      HL,(STOREB+1)
```

Listing 4 continues

Listing 4 continued

```
5300 220440           LD    (4004H),HL
5303 E1               POP   HL
5304 221B53           LD    (STOREB+1),HL
5307 C37344           JP    BASIC
530A F5       STARTB  PUSH  AF
530B C5               PUSH  BC
530C 3A8038   LOOPB   LD    A,(3880H)
530F A7               AND   A
5310 28FA             JR    Z,LOOPB
5312 010010           LD    BC,1000H          ;DELAY
5315 CD6000           CALL  0060H
5318 C1               POP   BC
5319 F1               POP   AF
531A C30A53   STOREB  JP    STARTB
              ;       CLEAN UP
              ;       SET POINTERS FOR BASIC
              ;
531D 214653   CLENUP  LD    HL,CLEND
5320 3A2501           LD    A,(0125H)
5323 FE49             CP    49H               ;CHECK IF MODEL III
5325 2004             JR    NZ,CLEN1
5327 22A740           LD    (40A7H),HL        ;BUFFER POINTER
532A 24               INC   H
532B 3600     CLEN1   LD    (HL),0
532D 23               INC   HL
532E 22A440           LD    (40A4H),HL        ;BASIC START
5331 227944           LD    (BAS),HL
5334 3600             LD    (HL),0
5336 23               INC   HL
5337 3600             LD    (HL),0
5339 23               INC   HL
533A 22F940           LD    (40F9H),HL
533D 22FB40           LD    (40FBH),HL
5340 22FD40           LD    (40FDH),HL
5343 C37344           JP    BASIC
5346 00       CLEND   DEFB  0
4473          BASIC   EQU   4473H
4479          BAS     EQU   4479H
43BE                  ORG   43BEH             ;SET COMMAND ADDRESSES
43BE 0652             DEFW  SABUBL
44F3                  ORG   44F3H
44F3 EE4F             DEFW  PATCH8
44FA                  ORG   44FAH
44FA 0250             DEFW  BOUNCE
4503                  ORG   4503H
4503 1450             DEFW  NOBOU
4508                  ORG   4508H
4508 BF52             DEFW  PATCHA
450E                  ORG   450EH
450E F952             DEFW  PATCHB
4516                  ORG   4516H
4516 994F             DEFW  DUALOF
451E                  ORG   451EH
451E 8B4F             DEFW  DUALON
4524                  ORG   4524H
4524 8B4F             DEFW  DUALON
452A                  ORG   452AH
452A CB4F             DEFW  PRDO
4530                  ORG   4530H
4530 D74F             DEFW  DOPR
4300                  END   4300H
```

have a printer you may want to keep the Dual command as a function. You would also have to eliminate this command as part of that set. PRDO is a very useful function and is well worth the memory used by all the route functions. The Dual N command turns off this function. The same precautions apply as for the Dual commands.

The DOPR command routes Print commands to a printer. As with PRDO, Dual N turns off this function and the same precautions apply. Only one command, Dual, PRDO, or DOPR, is enabled at a time—the last command entered.

Hex, octal, and decimal conversion is accomplished by the &H, &O, and &D commands, respectively. For example, executing PRINT &D255, PRINT &HFF, or PRINT &0377 return the value 255 to video. The &D function is especially useful for converting integers above 32767 to their signed value. For example, PRINT PEEK (&D40000) is equivalent to PRINT PEEK(−25536) and lets you avoid computing the negative value. The largest values allowed for the three functions are &HFFFF, &0177777, and &D65529

Well, there you have it! Many of the commands work in a slightly different fashion than intended by the original author. The Unikey function required extensive modification to work with the Model III as well as the Model I. The DEFUSR and USR functions are very loose derivatives of the program presented by the original author of the concept.

I applaud the original authors for their work. With this program, you will get a compilation of utility programs that I wish I had earlier on in my computing experience. ■

Tom Quindry, a Defense Department employee, lives at 6237 Windward Drive, Burke, VA 22015.

See List of Advertisers on Page 563

Storm Tracker

by Charles C. Williams

Forecast the landfall coordinates of an oncoming tropical storm or hurricane to see how much danger the storm poses to you.

This computer program forecasts landfall coordinates of an oncoming tropical storm or hurricane. The program supplements the hurricane watches and warnings issued by the weather bureau and gives you a clearer picture of just how much danger the storm poses to you.

You enter the coordinates of the storm, longitude and ln($\log_e$) latitude (the weather service furnishes these), as data into the program. The program computes a least-squares formula for the longitude and latitude and the latitude for a given longitude, and plots the curve points included in the computation and the predictive curve based on the input data.

The input value for M controls the depth of the data scan. When you are tracking a storm with the computer using the weather advisory's coordinates, you enter the longitude and ln($\log_e$) latitude data sets in reverse order. In the data stack the first data are on the bottom and the latest data at the top. For example, you would enter your data as follows:

995 Fifth data set
996 Fourth data set
997 Third data set
998 Second data set
999 First data set

The computer scans the data from the top down, line by line. Your value for M tells the computer how many data sets to read. When it reads M sets, it goes about its other chores.

With the data entered in reverse, and with a constant value for M (e.g., three), you can see that on the third day your computer will read the third, second, and first data sets. On the fourth day it will read the fourth, third, and second data sets. On the fifth day the data scan involves the fifth, fourth, and

```
1001 DATA 90.0, 3.295837
1002 DATA 90.5, 3.295837
1003 DATA 91.0, 3.277145
1004 DATA 91.2, 3.269569
1005 DATA 91.5, 3.265759
1006 DATA 92.0, 3.265759
1007 DATA 92.5, 3.265759
1008 DATA 93.5, 3.258097
1009 DATA 94.3, 3.254243
1010 DATA 95.4, 3.242592
1011 DATA 96.0, 3.218876
1012 DATA 96.6, 3.194583
1013 DATA 98.0, 3.178054
1014 DATA 97.5, 3.182212
1015 DATA 98.2, 3.161247
1016 DATA 98.5, 3.157000
1017 RETURN
1018 GOTO 1018
```

Fig. 1. Data for Anita

The Key Box

Models I and III
16K RAM
Cassette or Disk Basic

Program Listing

```
100 CLS:N=0:X=0:Y=0
101 PRINTTAB(12)":::HURRICANE TRACKING SYSTEM:::"
102 PRINTTAB(12)"**********   ALLEN   **********"
103 PRINTTAB(12)"LONGITUDE 'X'..LOG LATITUDE 'Y'"
104 PRINTTAB(12)" M = NUMBER OF (X,Y) DATA SETS "
105 PRINTTAB(12)"      CHARLES C WILLIAMS       "
106 PRINTTAB(12)"        COPYRIGHT 1980         "
107 INPUT "M=";M
108 CLS
109 READ X,Y
110 O=O+X
111 P=P+(X*X)
112 Q=Q+(X*Y)
113 S=S+Y
114 T=T+(Y*Y)
115 N=N+1
116 IFN=MTHEN117  ELSE109
117 U=((Q*M)-(O*S))/(M*M)
118 V=((P*M)-(O*O))/(M*M)
119 W=((T*M)-(S*S))/(M*M)
120 B=U/V
121 A=((S/N)-(B*O/N))
```

Listing continues

Listing continued

```
122 J=SQR(ABS(V)):K=SQR(ABS(W))
123 R=U/(J*K)
124 PRINTTAB(12)"CORRELATION COEFFICIENT (R) =";R
125 PRINT
126 PRINTTAB(12)"             ***   D   ***              "
127 PRINTTAB(12)"....VALUE(1)...PLOT(2)...CURVE(3)...."
128 INPUT "D=";D
129 IF D=1 THEN 132
13Ø IF D=2 THEN 135
131 IF D=3 THEN 144
132 INPUT "X=";X
133 PRINT,"LONGITUDE=";X;"    ";"LATITUDE=";EXP(A+(B*X))
134 GOTO 132
135 RESTORE:GOSUB 153
136 FOR N=1TOM
137 READ X, Y
138 Z=EXP(Y)
139 F=4Ø7-4*X
14Ø G= 91-3*Z
141 SET (F,G)
142 NEXT N
143 GOTO 184
144 RESTORE:GOSUB 153
145 FORX=98TO72 STEP-1
146 Z=EXP(A+(B*X))
147 F=4Ø7-(4*X)
148 G= 91-(3*Z)
149 IF(G<=1)OR(G>=47)THEN 152  ELSE 15Ø
15Ø SET (F,G)
151 NEXT X
152 GOTO 184
153 CLS:N=Ø:X=Ø
154 FOR Y=ØTO44 STEP 2
155 C=(3Ø-(N))
156 PRINT C"-"
157 N=N+1:IF N>=15 THEN 159  ELSE 154
158 NEXT Y
159 N=Ø:X=Ø:Y=Ø
16Ø FOR X=7TO119 STEP 8
161 Y=44
162 SET (X,Y)
163 NEXT X
164 N=Ø
165 FOR X=7TO119 STEP 8
166 C=(1ØØ-(2*N))
167 PRINT C;
168 N=N+1
169 NEXT X
17Ø PRINT@ 23,"NO";
171 PRINT@ 77,"GV";
172 PRINT@136,"CC";
173 PRINT@166,"TM";
174 PRINT@265,"BV";
175 PRINT@299,"MI";
176 PRINT@422,"HV";
177 PRINT@455,"TP";
178 PRINT@542,"*Y";
179 PRINT@651,"VC";
18Ø PRINT@662,"CP";
181 PRINT@766,"SD";
182 PRINT@817,"KG";
183 RETURN
184 GOTO 184
1ØØ1 DATA 87.Ø,3.Ø91Ø43
1ØØ2 DATA 87.9,3.1Ø9Ø61
1ØØ3 DATA 89.Ø,3.1Ø9Ø61
1ØØ4 DATA 89.6,3.113515
1ØØ5 DATA 9Ø.8,3.157ØØØ
1ØØ6 DATA 91.4,3.169686
1ØØ7 DATA 92.Ø,3.178Ø54
1ØØ8 DATA 92.8,3.194583
1ØØ9 DATA 93.1,3.194583
1Ø1Ø DATA 93.8,3.21Ø844
1Ø11 DATA 94.4,3.222868
1Ø12 DATA 95.2,3.23Ø8Ø4
1Ø13 DATA 95.5,3.226Ø44
1Ø14 DATA 96.1,3.23Ø8Ø4
1Ø15 DATA 96.5,3.238679
1Ø16 DATA 96.7,3.246491
1Ø17 DATA 96.9,3.254243
1Ø18 DATA 97.4,3.259569
1Ø19 DATA 98.1,3.292126
1Ø2Ø DATA 99.Ø,3.3Ø6887
1Ø21 END
```

```
1ØØ1 DATA 93.Ø,3.Ø68Ø53
1ØØ2 DATA 94.Ø,3.Ø44522
1ØØ3 DATA 94.Ø,3.Ø2Ø425
1ØØ4 DATA 94.3,2.995732
1ØØ5 DATA 94.Ø,3.Ø2Ø425
1ØØ6 DATA 95.Ø,3.Ø44522
1ØØ7 DATA 95.2,3.Ø587Ø7
1ØØ8 DATA 95.8,3.Ø587Ø7
1ØØ9 DATA 95.7,3.Ø77312
1Ø1Ø DATA 96.2,3.Ø63391
1Ø11 DATA 95.5,3.Ø68Ø53
1Ø12 DATA 95.Ø,3.Ø44522
1Ø13 DATA 94.5,3.Ø68Ø53
1Ø14 DATA 94.5,3.Ø68Ø53
1Ø15 DATA 94.Ø,3.Ø44522
1Ø16 DATA 93.5,3.Ø44522
1Ø17 DATA 93.Ø,3.Ø44522
1Ø18 DATA 92.5,3.Ø44522
1Ø19 DATA 93.Ø,3.Ø91Ø42
1Ø2Ø DATA 92.5,3.Ø91Ø42
1Ø21 DATA 92.5,3.Ø91Ø42
1Ø22 DATA 92.Ø,3.113515
1Ø23 DATA 89.5,3.238678
1Ø24 DATA 89.Ø,3.238678
1Ø25 DATA 88.Ø,3.258Ø97
1Ø26 DATA 87.5,3.269569
1Ø27 END
```

Fig. 2. Data for Henri

third sets. This gives you a moving scan with M data-set depth. You may vary the depth of the scan by changing the value for M. A large M gives a more generalized analysis. A smaller M makes the analysis more sensitive to path change.

After you enter a value for M, the computer asks for the selector value D. If you press 1 the computer then asks for a value of x, the longitude coordinate. The value for x is the longitude of the possible landfall area. For the western Gulf a value of 97 is suitable. The computer then determines a latitude value using the data you have input. This gives you the coordinates of the landfall based on the measurements made at that point in time.

Each point you determine is like a hit on a target. If the storm is well organized and consistent, you will have a tight group of points and a good idea of where the storm will landfall. If the storm is disorganized and erratic your group of points could spread all over. Even so each landfall point you determine is the best guess that can be made at that point in time and for the data input to the computer.

If you enter a value of 2 for D the computer plots the points that have been included by the value for M. If you enter a value of 3 for D the computer plots a total curve based on the input measurements.

The program also computes R, a measure of the correlation between the computer analysis and the actual data. R measures the goodness of fit and gives you an idea of how accurate your measurements are.

The Program Listing contains the complete program including data from hurricane Allen. Figures 1 and 2 contain the data for hurricanes Anita and Henri. If you want to analyze these hurricanes replace the data from lines 1001 and on in the program listing with the data from the figure for the hurricane you are interested in.

The computer analysis of Allen was hard to believe. At longitude 89, a little past Yucatan, the predictions had an 85 percent probability. By longitude 91 they had reached 90 percent probability. By longitude 94 the probability was 98 percent and rising. The latitude predicted at longitude 91.4 (25.8) was within .1 degree of the actual landfall coordinate (25.9) longitude 96.9. ■

Charles Williams can be reached at Number 14, 5320 Auden, Houston, TX 77005.

LOAD 80

6502 to Z80, Bit by Bit

by David S. Peckett

If you've ever wanted to convert those 6502 Assembly-language programs to run on a Z80 computer, here's a piece on how it's done.

One of the delights of working with computers is learning a particular system's programming language. To have any hope of adapting programs written for other systems, you must have a nodding acquaintance with other languages.

Even a widely used high-level language like Basic has as many dialects as there are computer types. Fortunately, if you know Level II Basic, you can usually read, say, Applesoft programs. That's one of the reasons for having high-level languages.

When you get down to the level of Assembly-language, things are not so easy. A thorough knowledge of the Z80 assembler is not much help when that special 6502 listing seems to be just what you need. The only way you can use it is to understand 6502 Assembly code well enough to work back to the program's algorithm. Then you can recode it for a Z80, and then debug it.

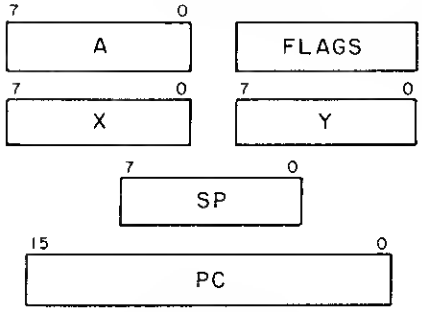


Fig. 1. 6502 Architecture

This article gives an easy way out. It describes a Level II Basic program that can translate 6502 assembler into equivalent Z80 code. If you only work in Basic, the program will be of no use; however, if you are a machine-code freak it could be a great help to you.

To keep things to a manageable size, the program translates a general 6502 listing to a general Z80 listing. Hardware-dependent sections are only translated as Assembly code—they are not altered to suit the architecture of the target computer.

For example, an Apple program may write directly to display memory; the translation would not automatically write to the TRS-80 display area. Similarly, you must make your own decisions about such items as where to put the stack.

Translation Approach

Before going into the detail of how the translation is handled, let's review the architectures of the 6502 and the Z80. Remember, a computer's architecture is reflected in its Assembly language. An understanding of one is essential to understand the other.

Figure 1 shows the 6502's basic configuration. The micro has a single 8-bit accumulator and a status register. It also has two 8-bit index registers (X and Y). Often interchangeable, these can be used for temporary data storage, as loop counters, and as indices for the micro's numerous addressing modes.

The 6502 also has an 8-bit stack pointer (SP) pointing to a stack permanently located on page 1 of memory, and a conventional 16-bit program counter (PC).

The Z80's architecture is shown in Fig. 2. The micro is more complex than the 6502 and has two sets of working registers. It can use either set at any time, but not together.

Each set has an 8-bit accumulator (A), with its associated flag register (F), and six general-purpose 8-bit reg-

The Key Box

Basic Level II
Model I
16K RAM
An editor/assembler

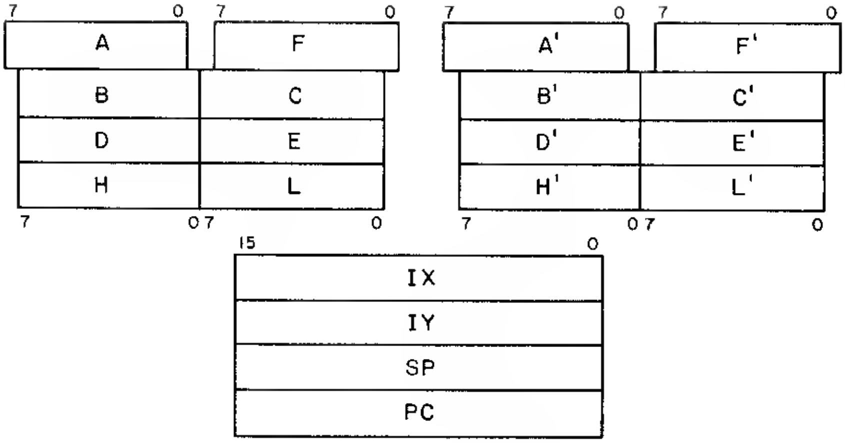


Fig. 2. Basic Z80 Architecture

isters (B–E,H,L). These six registers can be concatenated into three 16-bit registers (BC, DE, and Hl), to perform limited 16-bit arithmetic or to give pointers to anywhere in memory.

The Z80 also has a 16-bit stack pointer (SP), which means the stack can be anywhere in memory, and a 16-bit program counter (PC). The two 16-bit index registers (IX and IY) are redundant in this translation program.

Both microprocessors have an eight-bit accumulator which performs most arithmetic and both offer the same fundamental operations. The flags show the results of the operations, and both micros have equivalent carry, zero, and sign flags. Program counter operation is similar in the two micros, and the 6502's eight-bit SP is a subset of the Z80's 16-bit stack pointer.

Major Differences

Perhaps the most important difference between the two micros is in their approach to addressing memory. The 6502 has a wide selection of indexed, indirect, and compound addressing modes, including the more obvious immediate and absolute modes. The Z80 has only immediate and indirect operations—the latter use register HL as a pointer—and an indexing mode totally unlike the 6502.

The two sets of flags work differently. The 6502 sets them whenever it loads or manipulates A, X, or Y; the Z80 alters flags only after an arithmetic or logical operation. In subtraction and comparison operations, the 6502 sets its carry flag to show there was no borrow, while the Z80 clears the flag to show the same state. To add to the confusion, both micros treat their carry flags identically during addition.

Despite the differences, a Z80 can model a 6502 quite well. The grey areas (like the overflow flag) are unimportant. In fact, SED, OLD, BRK and BIT are the only four 6502 operations that the Z80 cannot copy.

Translation Approach

The program uses the Z80's registers to emulate those in the 6502. The accumulators, flag registers, stack pointers, and program counters are

See List of Advertisers on Page 563

equivalent and present no real problems. C and E in the Z80 model X and Y in the 6502. By keeping B and D at zero, we can reproduce the 6502's addressing modes.

The fundamental approach is to replace a single line of 6502 code with functionally equivalent Z80 code; any labels or comments are preserved. With operations like ADC or LDA this is straightforward, but some operations, such as CLV, are more complex.

Handling only one 6502 instruction at a time is not an ideal approach but was necessary to fit the program into 16K. With more RAM, it is not too difficult to generate a more intelligent program that could translate the sense

6502 Instruction Categories

Untranslatable:
BIT, BRK, CLD, SED

Single-Line, No Operand:
CLI, DEX, DEY, INX, INY, NOP, PHA, PHP, RTS, SEC, SEI

Single-Line, with Operand:
BCC, BCS, BEQ, BMI, BNE, BPL, BVC, BVS, JSR

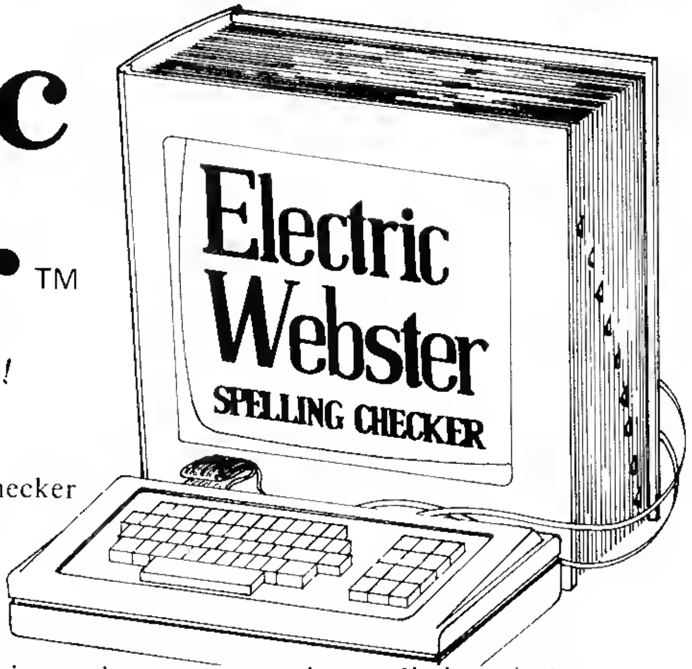
Standard Format:
ADC, AND, ASL, CMP, DEC, EOR, INC, JMP, LSR, ORA, ROL, ROR, STA, STX, STY

Irregular:
CLC, CLV, CPX, CPY, LDA, LDX, LDY, PLA, PLP, RTI, SBC, TAX, TAY, TSX, TXA, TXS, TYA

Table 1

Finally, a Spelling Checker that can SPELL!

Electric Webster™



The "Cadillac" of Spelling Checkers!
80 Microcomputing, 9/82

- ***FAST and ACCURATE*** — No other spelling checker comes close!
- ***INTEGRATED*** — Proofs and corrects from within all these popular word processing programs: Scripsit, Newscript, Lazy Writer, Electric Pencil, Superscript, and CopyArt.
- ***SMART*** — Finds and displays correct spellings instantly — no more clumsy dictionaries!
- ***HYPHENATES*** automatically — inserts discretionary hyphens with 100% accuracy. (optional)
- ***COMPLETE*** — One step proofing system with integrated Grammatical and Hyphenation features. (optional)

No other program can claim even one of these features.

EASY TO USE: Type your text using any of a number of popular word processing programs. When you are done, hit the appropriate key, and ELECTRIC WEBSTER proof-reads your document, displaying misspellings and typos on the screen.

Then, Correcting Electric Webster can display each error separately, requesting you to enter the correct spellings for each. You are given the options of displaying errors in context, adding words to ELECTRIC WEBSTER's 50,000 word dictionary, or even displaying the dictionary to find the correct spelling. If you think you know the correct spelling, Electric Webster will verify it for you.

Finally, Electric Webster CORRECTS YOUR TEXT, automatically inserts discretionary HYPHENS (optional), and points out GRAMMATICAL errors (optional), all with remarkable speed!

LOW PRICES: Standard Electric Webster is available for $89.50 (TRS-80™ Model I & III, or Apple™) or $149.50 (CP/M™, TRS-80™ Model II and all others.) The optional Correcting Feature can be added at any time ($60) as can Hyphenation ($50) and Grammatical ($40). During the closing months of 1982 only, we are offering 6 W/P integration programs FREE (reg. $35 each) with the purchase of Correcting Electric Webster.

REVIEWS OF MICROPROOF (EW's predecessor):

"There is simply no finer program available . . ."
Creative Computing, March 1982

"This is a very useful product and should be obtained by anyone who uses a word processor."
80 Microcomputing, August 1981

"The summary review of this program? One word—Excellent."
Computronics, September 1981

"In a comparative review of proofreading programs (with smaller dictionaries) MICROPROOF was found to be considerably faster than all the others, when tested against a 400 word sample document."
BYTE Magazine, November 1981

"A 1500 word document took 26 seconds to load, process and proof . . . it is very friendly and any prson able to use a word processing program can master it in moments."
InfoWorld, January 1982

"By far, the most capable and efficient of these spelling checker programs."
Microcomputing, June 1982

AND NOW ELECTRIC WEBSTER:

"Actually, Electric Webster is faster than its predecessor (Microproof) . . . and spelling corrections are immediately verified against the dictionary before being accepted . . ."
Microcomputing, June 1982

"Electric Webster is the Cadillac of vocabulary programs."
80 Microcomputing, September 1982

The Ultimate PROOFING SYSTEM

CORNUCOPIA SOFTWARE

Post Ofice Box 6111 Albany, California 94706 • (415) 524-8098

NOW AVAILABLE FOR APPLE!

45

of (small) blocks of 6502 assembler into Z80. Larger source programs than the 16K program's 1,500 byte buffer allows could also be handled.

Translation causes a growth in code size, so a translated program takes about three times as much RAM as the original and runs about one third slower. Neither penalty is significant.

Translating 6502 Operations

The fundamental problem is how to translate the 6502 operations. There is no room for all the details here, but the micro's instruction set can be divided into five general categories (see Table 1).

Untranslatable instructions are self-explanatory. The single-line commands replace a single 6502 operation with a single line of Z80 code. Single-line, no operand functions use implied addressing, while single-line with operand use a relative or absolute address. For example, the 6502 instructions DEY and BNE LOOP translate exactly as DEC E and JR NZ,LOOP.

The standard multi-line translations define a group in which the basic operation can be translated by a single line or modified by other lines controlling flags and the addressing mode. For example, EOR TEST becomes:

```
LD    HL,TEST
XOR   (HL)
```

These first three groups of translatable instructions can be handled by a simple look-up table approach, but the instructions in the last category do not

Program Listing 1

```
10 ' 6502 TO Z80 TRANSLATION PROGRAM
20 '(C) 1981, D S PECKETT
30 CLEAR 1000:DEFINT A-Z
40 DIM IN$(4),LN$(8),OD(57),TA$(8,4),TC(8),TV$(38)
500 'TOP-LEVEL SEGMENT FOR 6502-Z80 TRANSLATION
510 CLS: PRINT@272,"INITIALIZING SYSTEM";:GOSUB1000
520 GOSUB1500:'INPUT DATA
530 GOSUB3000:'PARSE A LINE
540 GOSUB5000:'TRANSLATE
550 IF F4 THEN 580:'END?
560 GOSUB2500:'SAVE TO TAPE
570 GOTO 530:'BACK FOR MORE
580 GOSUB9000:'TAIL END
590 PRINT@960,"TRANSLATION COMPLETE; REMOVE TAPE"
600 X$="":INPUT "DO YOU WANT TO DO ANOTHER TRANSLATION";X$:X$=LE
FT$(X$,1)
610 IF X$="N" END
620 X$="Y" THEN 520 ELSE GOTO 600
1000 'SYSTEM INITIALIZATION
1010 OC$="ADCANDASLBCCBCSBEQBITBMIBNEBPLBRKBVCBVSCLCCLDCLICLVCMP
CPXCPYDECDEXDEYENDEORINCINXINYJMPJSRLDALDXLDYLSRNOPORAPHAPHPPLAP
LPROLRORRTIRTSSBCSECSEDSEISTASTXSTYTAXTAYTSXTXATXSTYA":'6502 OPC
ODES
1020 FOR I=1 TO 57:READ X:OD(I)=X:NEXTI:'TRANSLATION CODES
1030 T$=CHR$(9):T1$="     ":C$=CHR$(13)
1040 FOR I=1 TO 35:READ X$:TV$(I)=X$:NEXT:'TRANSLATION VECTOR
1050 'FILL TRANSLATION ARRAY

1060 FOR I=1 TO 8:READ TC(I):FOR J=1 TO TC(I):READ X$:TA$(I,J)=X
$:NEXT J,I
1070 FOR I=32584 TO 32762:READ X:POKE I,X:NEXT I:'LOAD M/CODE
1080 POKE 16526,72:POKE 16527,127
1090 RETURN
1500 'INPUT DATA
1510 'READ IN LISTING TO BE TRANSLATED
1520 CLS
1530 PRINT "LOAD CASSETTE WITH 6502 PROGRAM INTO CASSETTE DRIVE
- HIT ANY   KEY WHEN READY"
1540 IF INKEY$="" THEN 1540
1550 A=USR(32588):'LOAD SOURCE CODE
1560 IF A=32588 PRINT "BAD TAPE - TRY AGAIN":GOTO 1530
1570 IF A=0 THEN PRINT"INPUT BUFFER FULL. ANY KEY TO CONTINUE TR
ANSLATION"ELSE GOTO 1590
1580 IF INKEY$="" THEN 1580
1590 L9=0:INPUT"START LINE NUMBER FOR OUTPUT";L9:L9=L9-10
1600 P9=30720:'POINTER FOR READING TEXT
1610 FD=0:FE=0:FF=0:FG=0:'ADD. MODE FLAGS
1620 'SET UP OUTPUT TAPE
1630 PRINT"LOAD CASSETTE WITH BLANK TAPE AND SET TO RECORD"
1640 X$="":INPUT"NAME OF OUTPUT FILE";X$:X$=X$+"      "
1650 X=VARPTR(X$)+1:POKE 32765,PEEK(VARPTR(X)):POKE 32766,PEEK(V
ARPTR(X)+1):'SET ADDRESS OF TITLE
1660 A=USR(32691):'LOAD HEADER
1670 LN=2:GOSUB2000
1680 OP$=LN$(1)+T1$+"LD   B,0"+C$+LN$(2)+T1$+"LD   D,0"+C$:'INIT
IALIZE REGISTERS
1690 GOSUB2500:RETURN
2000 'GENERATE LINE NUMBERS
2010 FOR IL=1 TO LN:L9=L9+10:L9$=STR$(L9):LN$(IL)=RIGHT$("0000"+
RIGHT$(L9$,LEN(L9$)-1)+" ",6):NEXT
2020 RETURN
2500 'OUTPUT TRANSLATION
2510 PRINT OP$;
2520 X=VARPTR(OP$):POKE 32765,PEEK(VARPTR(X)):POKE 32766,PEEK(VA
RPTR(X)+1):'OP$ ADDRESS
2530 A=USR(32720):'RECORD TRANS.
2540 RETURN
3000 'PARSE A LINE
3010 F1=0:F2=0:F3=0:IN$(1)="":IN$(2)="":IN$(3)="":IN$(4)="":'INI
TIALIZE VARIABLES
3020 IF CHR$(PEEK(P9))=";" THEN 3200:'COMMENT LINE?
3030 F9=0:I9U1
3040 X$=CHR$(PEEK(P9)):'READ A CHAR
3050 IF X$=C$ P9=P9+1:GOTO 3120:'FINISH ON CR
3060 IF X$=";" THEN F3=-1:'COMMENT FLAG
3070 IF (X$=" " OR X$=T$) AND NOT F9 AND NOT F3 AND I9<4 THEN I9
```

Listing 1 continues

have a regular pattern. Each opcode in this group must be handled individually. It's possible to rationalize in the case of closely related instructions like CPX and CPY, but that is all. As examples of translations of opcodes in this group, TAX becomes:

```
LD   C,A
INC  C    ;THESE 2 LINES..
DEC  C    ;..SET THE FLAGS
```

while CPY #$AB is translated as:

```
LD   D,A  ;SAVE A
LD   A,E  ;A=Y
CP   0AB
CCF       ;CY FLAG TO 6502 MODE
LD   A,D  ;RESTORE A
LD   D,B  ;D=0 AGAIN
```

These two examples illustrate the translation program's approach to controlling the flags. The INC, DEC in the first case sets the flags, and the second example's CCF keeps the Z80 carry flag in the state the 6502 would put it.

Translating 6502 Addressing Modes

There is no point in translating 6502 operations unless 6502 addressing modes can also be translated. The ab-

Listing 1 continued

```
=I9+1:F9=-1:P9=P9+1:GOTO 3040:'DELIMITER?
3080 IF (X$=" " OR X$=T$) AND F9 THEN P9=P9+1:GOTO 3040:'SAME DE
LIMITER?
3090 F9=0:'RESET FLAG
3100 IN$(I9)=IN$(I9)+X$:'READ A CHAR
3110 P9=P9+1:GOTO 3040:'NEXT
3120 'IN$(1-4) FILLED
3130 IF x(IN$(3),1)=";" AND IN$(4)="" IN$(4)=IN$(3):IN$(3)="":'C
ORRECT COMMENT FIELD
3140 'FILL RETURN STRINGS
3150 LB$=IN$(1)+T1$:O1$=IN$(2):O2$=IN$(3):CT$=T1$+IN$(4)
3160 GOSUB3500:IF F2 RETURN:'CHECK FOR ERRORS
3170 GOSUB4000:IF F2 RETURN:'FIND OPCODE
3180 GOSUB4500:RETURN:'CHECK ADDRESS MODE
3200 'COMMENT LINE
3210 CT$="":F1=-1
3220 X$=CHR$(PEEK(P9)):P9=P9+1:CT$=CT$+X$
3230 IF X$=C$ THEN RETURN ELSE GOTO 3220
3500 'CHECK FOR OBVIOUS SYNTAX ERRORS
3510 'LABEL FIELD
3520 IF LB$=T1$ THEN 3550
3530 IF LEN(LB$)>10 F2=-1:RETURN:'LABEL WRONG
3540 L=ASC(LEFT$(LB$,1)): IF NOT(LB$=T1$ OR (L>64ANDL<91)) F2=-1
:RETURN:'START OK?
3550 'OPERATION FIELD
3560 IF LEN(O1$)<>3 F2=-1:RETURN:'3 CHAR IN OP?
3570 L=ASC(LEFT$(O1$,1)):IF L<65 OR L>84 F2=-1:RETURN:'A-T START
?
3580 'COMMENT FIELD
3590 IF CT$=T1$ RETURN
3600 IF MID$(CT$,5,1)<>";" F2=-1:'NO ";" AT START
3610 RETURN
4000 'GET OPCODE
4010 IA=1:IB=57:FH=0
4020 IF FH OR IA>IB THEN 4060:'FINISHED?
4030 IM=(IA+IB)/2:AM$=MID$(OC$,IM*3-2,3)
4040 IF AM$=O1$ THEN FH=-1 ELSE IF AM$>O1$ THEN IB=IM-1 ELSE IA=
```

Listing 1 continues

See List of Advertisers on Page 563

solute and immediate modes are easy, as is the pure indirect mode; unfortunately, only JMP (Q) uses the indirect mode.

We also have to deal with the 6502's indexed, indirect-indexed and indexed-indirect modes; at the cost of a few lines of code, it is possible to imitate them all with a Z80.

The technique is to load register HL with the operand label (for example, Total from LDA (TOTAL,X)). This is followed by a CALL to one of four possible subroutines to modify HL to the intended address. Four subroutines (XIXX, IXIY, IXIND, and INDIX) emulate ABS,X, ABS,Y, (IND,X), and (IND),Y respectively. An example of this technique, EOR (STORE),Y, translates as:

```
        LD      HL,STORE
        CALL    INDIX
        XOR     (HL)
```

where:

```
INDIX EX   AF,AF'  ;SAVE A AND F
      LD   A,(HL)  ;LOW BYTE . . .
      INC  HL
      LD   H,(HL)  ;HIGH BYTE . . .
      LD   L,A     ;HL IS NOW POINTER
      ADD  HL,DE   ;ADD OFFSET
      EX   AF,AF'  ;RESTORE
      RET
```

When the 6502 program is translated, those subroutines actually called by the translation are added to the output.

Using the Translator

First, prepare the 6502 source code on tape. Use an editor/assembler to write the 6502 program in standard format, including line numbers, and save the program on tape in standard TRS-80 format. The translator accepts code in the standard 6502 assembler format as defined in Zaks' *Programming the 6502*; in addition, all comment fields must start with a semicolon. Do not include assembler pseudo-operations; since they vary from assembler to assembler, the translator does not attempt to handle them—they will be flagged as errors.

Reboot your computer and set memory size to 30720. You can then load the translation program and read the source tape into the 1,500-byte buffer. If the source program is in the wrong format, or is too long, you will get a warning message. In the latter case you will have to split the program into smaller parts. Even if the buffer overflows, you can run the translator immediately on what has been loaded.

Once the source code is in the buffer,

Listing 1 continued

```
IM+1
4050 GOTO 4020
4060 IF IA>IB THEN F2=-1:RETURN:'UNIDENTIFIED OPCODE
4070 IC=OO(IM):RETURN:'GET CODE
4500 'IDENTIFY ADDRESS MODE
4510 LN=LEN(O2$)
4520 IF O2$="" MA=1:RETURN:'IMPLIED
4530 IF O2$="A" MA=2:RETURN:'ACCUM
4540 L=ASC(LEFT$(O2$,1)):R=ASC(RIGHT$(O2$,1)):'1ST & LAST CHARS
4550 IF L<65 OR L>90 THEN 4620:'NOT ABS OR INDEXED
4560 IF LN<3 THEN 4580:'TOO SHORT FOR INDEXED
4570 IF MID$(O2$,LN-1,1)=","THEN 4600:'INDEXED?
4580 IF (R>47 AND R<58)OR(R>64 AND R<91)THEN MA=3 ELSE F2=-1:'AB
SOLUTE? GIVES "ABS" FOR "REL"
4590 RETURN
4600 IF R=88 THEN MA=4 ELSE IF R=89 THEN MA=5 ELSE F2=-1:'INDEXE
D
4610 O2$=LEFT$(O2$,LN-2):RETURN:'EXTRACT OPERAND
4620 IF L<>40 THEN 4680:'OPENING BRACKET?
4630 R$=RIGHT$(O2$,3)
4640 IF R$=",X)"THEN MA=6 ELSE IF R$="),Y" THEN MA=7 ELSE IF R=4
1 THEN MA=8 ELSE F2=-1:'(IND,X)/(IND),Y/(IND)?
4650 IF F2 RETURN
4660 IF MA=8 THEN O2$=MID$(O2$,2,LN-2) ELSE O2$=MID$(O2$,2,LN-4)
:'EXTRACT OPERAND
4670 RETURN
4680 'IMMEDIATE?
4690 IF L<>35 F2=-1:RETURN:'#?
4700 MA=9:L=ASC(MID$(O2$,2,1)):'SECOND CHAR
4710 IF L>47 AND L<58 THEN MI=1 ELSE IF L=36 THEN MI=2 ELSE IF L
=37 THEN MI=3 ELSE IF L>64 AND L<91 THEN MI=4 ELSE F2=-1:RETURN:
'IMMED. MODE?
4720 IF MI=1 OR MI=4 THEN O2$=RIGHT$(O2$,LN-1) ELSE O2$=RIGHT$(O
2$,LN-2):'EXTRACT OPERAND
4730 IF MI=2 O2$=O2$+"H":IF LEFT$(O2$,1)>"9" O2$="0"+O2$
4740 IF MI=3 O2$=O2$+"B"
4750 RETURN
5000 'TRANSLATION SUBROUTINE
5010 F4=0:'END FLAG
5020 IF NOT F1 THEN 5040
5030 LN=1:GOSUB2000:OP$=LN$(1)+CT$:RETURN:'COMMENT LINE
5040 IF F2 OR (MA=8 AND IC<>33) THEN 9500:'OBVIOUS ERROR
5050 'DATA OK - SELECT TRANSLATION MODE
5060 IF IC=25 F4=-1:RETURN:'END
5070 IF IC<5 THEN GOSUB5200 ELSE IF IC>4 AND IC<16 THEN GOSUB530
0 ELSE IF IC>15 AND IC<25 GOSUB5400
5080 IF IC>25 AND IC<41 THEN GOSUB5600 ELSE IF IC>40 AND IC<49 T
HEN GOSUB5800 ELSE IF IC>48 GOSUB6000
5090 RETURN:'TRANSLATION OVER
5200 'UNTRANSLATABLE LINES
5210 LN=2:GOSUB2000:OP$=LN$(1)+";***THE NEXT OPCODE IS UNTRANSLA
TABLE***"+C$+LN$(2)+IN$(1)+T1$+IN$(2)+T1$+IN$(3)+T1$+IN$(4)+C$:R
ETURN
5300 'CLI-SEI
5310 IF MA<>1 THEN 9500:'IMM. MODE?
5320 LN=1:GOSUB2000:OP$=LN$(1)+LB$+TV$(IC-4)+CT$+C$:RETURN
5400 'BCC-JSR
5410 IF MA<>3 THEN 9500:'ABS. MODE?
5420 LN=1:GOSUB2000:OP$=LN$(1)+LB$+TV$(IC-4)
5430 IF IC<24 THEN OP$=OP$+"," ELSE OP$=OP$+T1$:'BRANCHES
5440 OP$=OP$+O2$+CT$+C$:RETURN
5600 'STND MULTI-MODE FORMAT ADC-STY
5610 IF MA=1 OR (MA=2 AND NOT(IC=28ORIC=34ORIC=36ORIC=37)) OR (M
A=9 AND NOT(IC=26ORIC=27ORIC=29ORIC=31ORIC=35)) THEN 9500:'BAD S
YNTAX
5620 LN=1:GOSUB2000
5630 IF MA=2 OR MA=9 OP$=LN$(1)+LB$+TV$(IC-5)+O2$+CT$+C$:GOTO 57
00:'IMMED OR ACCUM
5640 IF MA=3 OR MA=8 OP$=LN$(1)+LB$+"LD   HL,"+O2$+CT$+C$:'ABS O
R (IND)
5650 IF MA>3 AND MA<8 GOSUB8000:'INDEXED
5660 LN=1:GOSUB2000
5670 OP$=OP$+LN$(1)+T1$+TV$(IC-5)+"(HL)"
5680 IF O1$="STA" THEN O$=",A" ELSE IF O1$="STX" THEN O$=",C" EL
SE IF O1$="STY" THEN O$=",E" ELSE O$="":'SORT OUT "STORES"
5690 OP$=OP$+O$+C$
5700 IF O1$="CMP" LN=1:GOSUB2000:OP$=OP$+LN$(1)+T1$+"CCF"+C$:'CO
NTROL CY FLAG
5710 RETURN
5800 'IRREGULAR SHORT
5810 IF MA<>1 THEN 9500:'NO OPERAND
5820 ID=IC-40:'OPCODE
5830 LN=TC(ID):GOSUB2000:OP$=LN$(1)+LB$+TA$(ID,1)+CT$+C$
5840 FOR I=2 TO TC(ID):OP$=OP$+LN$(I)+T1$+TA$(ID,I)+C$:NEXT I
```

Listing 1 continues

Listing 1 continued

```
5850 RETURN
6000 'IRREGULARS
6010 ON (IC-48) GOSUB 6200,6400,6400,6600,6800,6800,7000,7200,72
00
6020 RETURN
6200 'CLV
6210 IF MA<>1 THEN 9500:'OPERAND?
6220 LN=8:GOSUB2000
6230 OP$=LN$(1)+LB$+"LD    D,C"+CT$+C$+LN$(2)+T1$+"PUSH    AF"+C$+
LN$(3)+T1$+"POP    BC"+C$+LN$(4)+T1$+"RES    2,C"+C$+LN$(5)+T1$+"P
USH    BC"+C$
6240 OP$=OP$+LN$(6)+T1$+"POP    AF"+C$+LN$(7)+T1$+"LD    C,D"+C$+L
N$(8)+T1$+"LD    B,0"+C$
6250 RETURN
6400 'CPX,CPY
6410 IF NOT(MA=3 OR MA=9) THEN 9500
6420 P$="B":Q$="C":R$="D":IF IC=51 P$="D":Q$="E":R$="B":'CPX OR
CPY?
6430 LN=2:GOSUB2000:OP$=LN$(1)+LB$+"LD    "+P$+",A"+CT$+C$+LN$(2)
+T1$+"LD    A,"+Q$+C$
6440 IF MA=9 THEN 6470:'IMMEDIATE
6450 LN=2:GOSUB2000:OP$=OP$+LN$(1)+T1$+"LD    HL,"+O2$+C$+LN$(2)+
T1$+"CP    (HL)"+C$
6460 GOTO 6480
6470 LN=1:GOSUB2000:OP$=OP$+LN$(1)+T1$+"CP    "+O2$+C$
6480 LN=3:GOSUB2000:OP$=OP$+LN$(1)+T1$+"LD    A,"+P$+C$+LN$(2)+T1
$+"LD    "+P$+","+R$+C$+LN$(3)+T1$+"CCF"+C$
6490 RETURN
6600 'LDA
6610 IF MA<3 THEN 9500:'BAD ADD. MODE?
6620 LN=1:GOSUB2000
6630 IF MA=3 THEN OP$=LN$(1)+LB$+"LD    A,("+O2$+")"+CT$+C$:GOTO
6670:'ABSOLUTE?
6640 IF MA=9 THEN OP$=LN$(1)+LB$+"LD    A,"+O2$+CT$+C$:GOTO 6670:
'IMMED?
6650 GOSUB8000:'INDEXING
6660 LN=1:GOSUB2000:OP$=OP$+LN$(1)+T1$+"LD    A,(HL)"+C$
6670 LN=2:GOSUB2000:OP$=OP$+LN$(1)+T1$+"INC    A"+C$+LN$(2)+T1$+"
DEC    A"+C$:'SET FLAGS
6680 RETURN
6800 'LDX/Y
6810 P$="C":IF IC=54 P$="E"
6820 IF MA<3 THEN 9500:'BAD ADD MODE?
6830 LN=1:GOSUB2000
6840 IF MA=9 OP$=LN$(1)+LB$+"LD    "+P$+","+O2$+CT$+C$:GOTO 6870:
'IMMED.
6850 IF MA=3 THEN OP$=LN$(1)+LB$+"LD    HL,"+O2$+CT$+C$ ELSE GOSU
B8000:'ABS.
6860 LN=1:GOSUB2000:OP$=OP$+LN$(1)+T1$+"LD    "+P$+",(HL)"+C$
6870 LN=2:GOSUB2000:OP$=OP$+LN$(1)+T1$+"INC    "+P$+C$+LN$(2)+T1$
+"DEC    "+P$+C$
6880 RETURN
7000 'SBC
7010 IF MA<3 THEN 9500:'BAD ADD MODE?
7020 IF MA<>9 THEN 7040
7030 LN=3:GOSUB2000:OP$=LN$(1)+LB$+"CCF"+CT$+C$+LN$(2)+T1$+"SBC
  A,"+O2$+C$+LN$(3)+T1$+"CCF"+C$:RETURN
7040 LN=1:GOSUB2000
7050 IF MA=3 OP$=LN$(1)+LB$+"LD    HL,"+O2$+CT$+C$:GOTO 7070
7060 GOSUB8000:'INDEXING
7070 LN=3:GOSUB2000:OP$=OP$+LN$(1)+T1$+"CCF"+C$+LN$(2)+T1$+"SBC
  A,(HL)"+C$+LN$(3)+T1$+"CCF"+C$:RETURN
7200 'TSX/TXS
7210 IF MA<>1 THEN 9500:'BAD ADD. MODE?
7220 LN=3:GOSUB2000:OP$=LN$(1)+LB$+"LD    H,B"+CT$+C$+LN$(2)+T1$+
"LD    L,B"+C$+LN$(3)+T1$+"ADD    HL,SP"+C$:IF IC=56 THEN 7240
7230 LN=2:GOSUB2000:OP$=OP$+LN$(1)+T1$+"LD    L,C"+C$+LN$(2)+T1$+
"LD    SP,HL"+C$:IF IC=57 THEN 7250
7240 LN=1:GOSUB2000:OP$=OP$+LN$(1)+T1$+"LD    C,L"+C$
7250 LN=2:GOSUB2000:OP$=OP$+LN$(1)+T1$+"INC    C"+C$+LN$(2)+T1$+"
DEC    C"+C$
7260 RETURN
8000 'INDEXED MODE
8010 OP$=LN$(1)+LB$+"LD    HL,"+O2$+CT$+C$:'INITIALIZE HL
8020 LN=1:GOSUB2000:OP$=OP$+LN$(1)+T1$+"CALL    "
8030 IF MA=4 OP$=OP$+"IXIX":FD=-1:'IND,X
8040 IF MA=5 OP$=OP$+"IXIY":FE=-1:'IND,Y
8050 IF MA=6 OP$=OP$+"IXIND":FF=-1:'(IND,X)
8060 IF MA=7 OP$=OP$+"INDIX":FG=-1:'(IND),Y
8070 OP$=OP$+C$:'FINISH OFF
8080 RETURN
9000 'WRAPUP ROUTINE
9010 IF FD THEN LN=2:GOSUB2000:OP$=LN$(1)+"IXIX"+T1$+"EX    AF,AF
```

Listing 1 continues

follow the program's instructions. It will call for a name for the output tape, and the line number at which the output must start. The program automatically increments each output line number by 10.

Error Detection

The translator does a fair amount of syntax checking on the 6502 program, but it only looks for errors that might confuse it. Examples are comments not starting with a semicolon; an operation field not having three characters; more than six characters in a label; and non-label and non-immediate operand fields. Some errors slip through, but they should be caught by the Z80 assembler after translation.

Manual Tuning

Once the 6502 program has been translated, a little manual fine-tuning can improve output efficiency. If the translated program has to handle interrupts several manual additions must be made.

Whenever a 6502 operation sets the flags, but the corresponding Z80 instruction does not, the translator adds:

```
INC n
DEC n
```

where n is A, C or E. This puts the flags into the correct state.

The next instructions often do not need the flags. Examine the translation and, if the flags are redundant, delete the INC/DEC pairs.

Listing continued

```
'"+C$+LN$(2)+T1$+"ADD   HL,BC"+C$:GOSUB9200:'IXIX?
9020 IF FE THEN LN=2:GOSUB2000:OP$=LN$(1)+"IXIY"+T1$+"EX   AF,AF
'"+C$+LN$(2)+T1$+"ADD   HL,DE"+C$:GOSUB9200:'IXIY?
9030 IF FF THEN LN=6:GOSUB2000:OP$=LN$(1)+"IXIND"+T1$+"EX   AF,A
F'"+C$+LN$(2)+T1$+"ADD   HL,BC"+C$+LN$(3)+T1$+"LD   A,(HL)"MC$+L
N$(4)+T1$+"INC   HL"+C$+LN$(5)+T1$+"LD   H,(HL)"+C$+LN$(6)+T1$+"
LD   L,A"+C$:GOSUB9200:'IXIND?
9040 IF FG THEN LN=6:GOSUB2000:OP$=LN$(1)+"INDIX"+T1$+"EX   AF,A
F'"+C$+LN$(2)+T1$+"LD   A,(HL)"+C$+LN$(3)+T1$+"INC   HL"+C$+LN$(
4)+T1$+"LD   H,(HL)"+C$+LN$(5)+T1$+"LD   L,A"+C$+LN$(6)+T1$+"ADD
   HL,DE"+C$:GOSUB9200:'IXIND?
9050 LN=1:GOSUB2000:OP$=LN$(1)+T1$+"END"+C$+CHR$(26):GOSUB2500:'
FINAL LINE
9060 A=USR(504):'TAPE OFF
9070 RETURN
9200 'ALL "IX"S END SAME WAY
9210 LN=2:GOSUB2000:OP$=OP$+LN$(1)+T1$+"EX   AF,AF'"+C$+LN$(2)+T
1$+"RET"+C$:GOSUB2500:RETURN
9500 'SYNTAX ERROR
9510 LN=2:GOSUB2000:OP$=LN$(1)+";*** SYNTAX ERROR BELOW ***"+C$+
LN$(2)+IN$(1)+T1$+IN$(2)+T1$+IN$(3)+T1$+IN$(4)MC$:RETURN
10000 'DATA FOR OPCODE TRANSLATION
10010 DATA 26,27,28,16,17,18,1,19,20,21,2,22,23,41,3,5,49,29,50,
51,30,6,7,25,31,32,8,9,33,24,52,53,54,34,10,35,11,12,42,43,36,37
,44,13,55,14,4,15,38,39,40.45,46,56,47,57,48
10020 'DATA FOR TRANSLATION VECTOR
10030 'ONE-LINERS
10040 DATA EI,DEC   C,DEC   E,INC   C,INC   E,NOP,PUSH   AF,PUSH
   AF,RET,SCF,DI
10050 'JUMPS
10060 DATA JR   NC,JR   C,JR   Z,JP   M,JR   NZ,JP   P,JP
   PE,JP   PO,CALL
10070 'STANDARDS
10080 DATA "ADC   A,",AND   ,SLA   ,CP   ,DEC   ,XOR   ,INC   ,J
P   ,SRL   ,OR   ,RL   ,RR   ,LD   ,LD   ,LD
10090 'DATA FOR TRANSLATION ARRAY
10100 DATA 2.SCF,CCF,4,POP   HL,"LD   A,H",INC   A,DEC   A,3,"LD
   H,A",POP   AF,"LD   A,H",4,"LD   H,A",POP   AF,"LD   A,H",RET
I
10110 DATA 3,"LD   C,A",INC   A,DEC   A,3,"LD   E,A",INC   A,DEC
   A,3,"LD   A,C",INC   A,DEC   A,3,"LD   A,E",INC   A,DEC   A
10120 'M/CODE DATA
10130 'START
10140 DATA 205,127,10,233
10150 'INPUT
10160 DATA 253,33,63,60,175,205,18,2,205,150,2,205,53,2,254,211,
40,6,205,248,1,195,154,10,6,6,205,53,2,16,251,33,0,120,17,8,7,6,
6,205,53,2.254,26,40.31,16,247,205,53,2,119,35,27,254,13,32,246,
62,42,253.190,0,32,2,62.32,253,119,0,203,122,40
10170 DATA 219,17,0,0,205,248,1,62,9,119,35,62,69,119,35,62,78,1
19,35,62,68,119,35,62.13,119,235,195,154,10
10180 'HEADER
10190 DATA 42.253,127,94,35,86,235,175,205,18,2,205,135,2,62,211
,205,100,2,6,6,126,205,100,2,35,16,249,201
10200 'OUTPUT
10210 DATA 42.253,127,70,35,94,35,86,235,14,5,126,254,26,32,4,20
5,100,2.201,203,255,205,100,2,35,5,13,32,237,126,205,100,2,35,5,
200,254,13,40,224,24,243
```

Listing continues

Because of the difference between the two micros' carry flags, the program has a CCF after each comparison operation. If the flag's state is irrelevant, delete the line. Be cautious—the carry flag may be used later. If in doubt, leave it in.

A third way of simplifying the translation is found in some addition and subtraction operations. The 6502 always takes the carry into account in these operations and has to achieve an add without carry by code like:

```
CLC
ADC TOTAL
```

which is translated as:

```
SCF
CCF
LD   HL,TOTAL
ADC  (HL)
```

and can be simplified to:

```
LD   HL,TOTAL
ADD  (HL)
```

Interrupt Handling

Other than by the presence of RTIs, the translator cannot tell that a 6502 program is interrupt driven. The translation must be modified before it will work. Add to the start of the program:

```
IM1     ;THIS MODE EMULATES 6502
         INTERRUPTS
        ;SUITABLE CODE TO DEFINE THE
         INTERRUPT VECTOR
```

The 6502 saves the program counter and the flags in the stack during an interrupt, while the Z80 only preserves the program counter. Whenever an interrupt handler could affect the flags add PUSH AF to its start and POP AF before the RETI.

Assembler Directives

The assembler directives will be different from the originals. In addition, the stack area must be defined with a LD SP,wxyz; since the translation is set up to use a stack on a single page of memory, wxyz would be best as wx00.

For an example of the translator in action, see Program Listings 2 and 3. Listing 2 is a segment of 6502 code with several deliberate errors. Listing 3 is an unedited copy of its translation. All errors have been highlighted, and an error-free translation works perfectly.

Listing 3 also shows lines which could be edited. For instance, the INC/DEC in lines 1210 and 1220 is redundant; the instructions could safely be removed. On the other hand, the CCF in line 1410, after the compari-

Line No.	16K	32K	48K
30	1000	1500	1500
1070	32584	−16568	−184
1070	32762	−16390	−6
1080	127	191	255
1550, 1560	32588	−16564	−180
1600	P9=30720	P9!=31232	P9!=31232
1650	32765	−16387	−3
1650	32766	−16386	−2
1660	32691	−16461	−77
2520	32765	−16387	−3
2520	32766	−16386	−2
2530	32720	−16432	−48
3020, 3040	P9	Note 1	Note 1
3050, 3070 3080, 3110	P9	P9!	P9!
3220	P9	Note 1	Note 1
10160	120 (34th item)	122	122
10160	7 (37th item)	69	133
10190	127	191	255
10210	127	191	255

Note 1:
In these lines, P9 should be replaced by:
P9!+(P9!>Z1)*Z2!
and an extra line should be inserted into the program
1085 Z1=32767:Z2!=65536

Note 2:
Set the memory size to 31220 in the 32K and 48K versions.

Table 2. Conversion for 32K and 48K

```
00100 ;SUBROUTINE TO SAVE DATA HELD IN STACK
00110 ;SP TO THE DATA IS PASSED IN 'BASE'
00120 THIS IS A BAD COMMENT LINE
00130 SAVE     LDX     BASE     ;RESET SP TO..
00140          TXS              ;..BASE OF DATA
00150          TYA              ;DOUBLE COUNT..
00160          ASL     A        ;..FOR STORAGE
00170          TAX              ;SET INDEX
00180 NXTBYTT  PLA              ;BAD LABEL
00190          TAY              ;SAVE BYTE
00200          AND     #$0F     ;TRUNCATE
00210          JSR     ASCII    ;CONVERT
00220          TYY              ;BAD OPCODE
00230          LSR     A
00240 ;PLUS 3 MORE 'LSR A'S
00250          JSR     ASCII    BAD COMMENT FIELD
00260          BNE     NXTBYT   ;FINISHED?
00270          RTS              ;YES, RETURN
00280 ;
00290 ASCII    ORA     #$30     ;ASCII PREFIX
00300          CMP     #$3A     ;TEST TO SEE..
00310          BCC     NUMBER   ;..IF A-F
00320          ADC     #6       ;ALPHA OFFSET
00330 NUMBER   STA     DATA-1,X   ;SAVE CHARACTER
00340          DEX              ;NEXT STORE
00350          RTS
00360          END
```

Program Listing 2

son, is vital because the next instruction tests the carry flag.

The translation shows the address-mode subroutine IXIX was needed to handle the STA DATA-1,X at line 330 of Listing 2. The other indexing subroutines were not added since they were not required.

Limitations

A translation program of this kind has limitations. None are serious, but be aware of them:

- The operand must be a label or an immediate quantity; you cannot use LDA $12B7 or similar expressions.
- The 6502 can handle binary-coded decimal (BCD) quantities directly. It was not possible to translate this within the 16K size limitation.
- Because Z80 stack operations move two bytes at a time, compared with the 6502's single-byte operations, there can only be a maximum of 128 items on the stack at any time. Also, manipulating data in the stack may cause corruption.
- The overflow flag cannot always be trusted.
- Timing loops must be readjusted.

The translator can handle a fuller and more rational set of addressing modes than the 6502 actually provides. As far as the program is concerned, any address memory instruction can use any mode other than implied, which eliminates the 6502's arbitrary restrictions on what modes can be used when. Thus you can write pseudo-6502 code that makes the fullest possible use of the micro's unusually complete set of addressing options.

```
01000     LD   B,0
01010     LD   D,0
01020 ;SUBROUTINE TO SAVE DATA HELD IN STACK
01030 ;SP TO THE DATA IS PASSED IN 'BASE'
01040 ;*** SYNTAX ERROR BELOW ***
01050 THIS    IS    A    BAD COMMENT LINE
01060 SAVE    LD   HL,BASE    ;RESET SP TO..
01070     LD   C,(HL)
01080     INC  C
01090     DEC  C
01100     LD   H,B    ;..BASE OF DATA
01110     LD   L,B
01120     ADD  HL,SP
01130     LD   L,C
01140     LD   SP,HL
01150     INC  C
01160     DEC  C
01170     LD   A,E    ;DOUBLE COUNT..
01180     INC  A
01190     DEC  A
01200     SLA  A    ;..FOR STORAGE
01210     LD   C,A    ;SET INDEX
01220     INC  A
01230     DEC  A
01240 ;*** SYNTAX ERROR BELOW ***
01250 NXTBYTT    PLA        ;BAD LABEL
01260     LD   E,A    ;SAVE BYTE
01270     INC  A
01280     DEC  A
01290     AND  0FH    ;TRUNCATE
01300     CALL   ASCII    ;CONVERT
01310 ;*** SYNTAX ERROR BELOW ***
01320     TYY        ;BAD OPCODE
01330     SRL  A
01340 ;PLUS 3 MORE 'LSR A'S
01350 ;*** SYNTAX ERROR BELOW ***
01360     JSR    ASCII    BAD COMMENT FIELD
01370     JR     NZ,NXTBYT    ;FINISHED?
01380     RET    ;YES, RETURN
01390 ;
01400 ASCII    OR   30H    ;ASCII PREFIX
01410     CP   3AH    ;TEST TO SEE..
01420     CCF
01430     JR   NC,NUMBER    ;..IF A-F
01440     ADC  A,6    ;ALPHA OFFSET
01450 NUMBER    LD    HL,DATA-1    ;SAVE CHARACTER
01460     CALL   IXIX
01470     LD   (HL),A
01480     DEC  C    ;NEXT STORE
01490     RET
01500 IXIX    EX   AF,AF'
01510     ADD  HL,BC
01520     EX   AF,AF'
01530     RET
01540     END
```

Program Listing 3

Extending the Program

The program would run perfectly in the bottom 16K of a 32K or 48K TRS-80, and it is easily modified to make use of the extra space. The modification enlarges the size of the input buffer to allow the program to translate larger 6502 programs.

Table 2 shows the changes you should make to match the program to the larger computers; the 32K expansion would give more than enough space for any likely source code.

Do not forget that, even with the larger memory, this will still be a cassette-based program.

Conclusion

This program is a useful tool for translating 6502 programs to run on Z80-based microcomputers. It is written for a TRS-80 Model I, but with a lot of effort, it could be adapted to other systems.

The program's output is less efficient than the original code, but the translation is adequate for most purposes. The program's main weakness is that, since it has to fit into a 16K computer, it is not as intelligent as it might be. ■

David Peckett, a chartered electrical engineer, can be reached at 1 Delapoer Drive, Haverford West, Dyfed SA61 1HZ, Wales, UK.

✓See List of Advertisers on Page 563

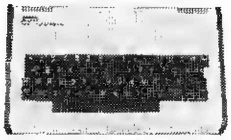
JE600 Hexadecimal Encoder Kit

FULL 8-BIT LATCHED OUTPUT 19-KEY KEYBOARD

The JE600 Encoder Keyboard Kit provides two separate hexadecimal digits produced from sequential key entries to allow direct programming for 8-bit microprocessor or 8 bit memory circuits. Three additional keys are provided for user operations with one having a bistable output available. The outputs are latched and monitored with 9 LED readouts. Also included is a key entry strobe. Features: Full 8-bit latched output for microprocessor use. Three user-define keys with one being bistable operation. Debounce circuit provided for all 19 keys. 9 LED readouts to verify entries. Easy interfacing with standard 16-pin IC connector. Only +5VDC required for operation. Size: 3½"H x 8½"W x 8¾"D

JE600/DTE-HK (After assembled as pictured above) $99.95
JE600 Kit 19-Key Hexadec. Keyboard, PC Board & Cmpnts. (no case) .. $59.95
K19 19-Key Keyboard (Keyboard only) $14.95
DTE-HK (case only 3½"Hx8½"Wx8¾"D) $44.95

JE610 ASCII Encoded Keyboard Kit



The JE610 ASCII Keyboard Kit can be interfaced into most any computer system. The kit comes complete with an industrial grade keyboard switch assembly (62-keys), IC's, sockets, connector, electronic components and a double sided printed wiring board. The keyboard assembly requires +5V @ 150mA and −12V @ 10 mA for operation. Features: 60 keys generate the 126 characters, upper and lower case ASCII set. Fully buffered. Two user-define keys provided for custom applications. Caps lock for upper-case-only alpha characters. Utilizes a 2376 (40-pin) encoder read-only memory chip. Outputs directly compatible with TTL/DTL or MOS logic arrays. Easy interfacing with a 16-pin dip or 18-pin edge connector. Size: 3¾"H x 14½"W x 8¾"D

JE610/DTE-AK (After assembled as pictured above) ... $124.95
JE610 Kit 62-Key Keyboard, PC Board, & Components (no case) .... $ 79.95
K62 62-Key Keyboard (Keyboard only) .. $ 34.95
DTE-AK (case only — 3¾"Hx1½"Wx8¾"D) $ 49.95

NEW! JE212 — Negative 12VDC Adapter Board Kit for JE610 ASCII KEYBOARD KIT Kit/ Provides -12VDC from incoming 5VDC . $9.95

JE215 Adjustable Dual Power Supply

General Description: The JE215 is a Dual Power Supply with independent adjustable positive and negative output voltages. A separate adjustment for each of the supplies provides the user unlimited applications for IC current voltage requirements. The supply can also be used as a general all-purpose variable power supply.

FEATURES:
- Adjustable regulated power supplies, pos. and neg. 1.2VDC to 15VDC.
- Power Output (each supply): 5VDC @ 500mA, 10VDC @ 750mA, 12VDC @ 500mA, and 15VDC @ 175mA.
- Two, 3-terminal adj IC regulators with thermal overload protection.
- Heat sink regulator cooling
- LED "on" indicator
- Printed Board Construction
- 120VAC input
- Size: 3-1/2"w x 5-1/16"L x 2"H

JE215 Adj. Dual Power Supply Kit (as shown) . . $24.95
(Picture not shown but similar in construction to above)
JE200 Reg. Power Supply Kit (5VDC, 1 amp) . . $14.95
JE205 Adapter Brd. (to JE200) ±5,±9 & ±12V. . $12.95
JE210 Ver. Pwr. Sply. Kit, 5-15VDC, to 1.5amp. . $19.95

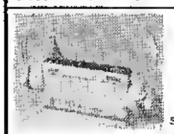
HP·Display Sale·National

5082 Series — 0.43 inch — 7-Segment

Part Number	Color	Description	1-3 Price	SALE PRICE
5082-7650	Hi Eff Red	CA - LHD	.99	4/$2.49
5082-7651	Hi Eff Red	CA - RHD	.99	4/$2.49
5082-7653	Hi Eff Red	CC - RHD	.99	4/$2.49
5082-7656	Hi Eff Red	Overflow ± 1RHD	.99	4/$2.49
5082-7660	Yellow	CA - LHD	.99	4/$2.49
5082-7661	Yellow	CA - RHD	.99	4/$2.49
5082-7663	Yellow	CC - RHD	.99	4/$2.49
5082-7670	Green	CA - LHD	.99	4/$2.49
5082-7671	Green	CA - RHD	.99	4/$2.49
5082-7673	Green	CC - RHD	.99	4/$2.49
5082-7676	Green	Overflow ± 1RHD	.99	4/$2.49
5082-7750	Red	CA - LHD	.99	4/$2.49
5082-7751	Red	CA - RHD	.99	4/$2.49
5082-7756	Red	Overflow ± 1RHD	.99	4/$2.49
5082-7760	Red	CC - RHD	.99	4/$2.49

CA-Comm. Anode CC-Comm. Cathode LHD/RHD-Left/right hand dec.

UTIC Mini Stereo AM/FM Receiver



CLOSE-OUT!

WITH HEADPHONES For Joggers, Cyclists, Skaters & Sports Events

FEATURES: Lightweight headphones. Left/right balance control. Full fidelity stereo sound. Additional black soft carrying case & shoulder strap. Belt clip (hands free). Operates on 3 AA cell batteries (not incl.). Compact size: 3⅞" x 4⅞" x 1". Wt. 6 oz.

Model 2830 . . . . $29.95

KEYBOARDS — POWER SUPPLIES

MICRO SWITCH 69-KEY KEYBOARD
Data Entry Keyboard, Encoded Output: 8-bit Parallel EBC DIC. Switching Hall Effect, 24-pin Edge Card Connection. Complete with Pin Connection.
15⅝"L x 5¼"W x 1⅜"H
Part No. KB69SD12-2 (Fits Into DTE-20 Enclosure) . . . . $19.95 each

DATANETICS 74-KEY KEYBOARD
ASCII Encoded Keyboard, Output: Even Parity ASCII, Supply voltage +5, -12 volt. Switching: Mechanical SPST — 50-pin Connection. Complete with Pin Connection.
16¾"L x 5½"W x 1⅜"H
Part No. KB354 (Fits Into DTE-20 Enclosure) . . . . $29.95 each

MICRO SWITCH 85-KEY KEYBOARD
Word Processing Keyboard, 26 Pin Edge Card Connection. Supply Voltage +5VDC. Main Keyboard is QWERTY. Additional Key Pads for Cursor and word processing functions.
23"Lx5¾"Wx1-3/8"H
Part No. 85SD18-1 . . . . $29.95 each

MICRO SWITCH 88-KEY KEYBOARD (PARALLEL)
Data Entry Keyboard used in a Diablo 1640 Terminal. Supply Voltage: +5V, -12V. Switching: Hall Effect — 10-pin Edge Card Connection. Schematic included. Uses 8048 Encoder Chip.
18¼"L x 5¾"W x 1⅝"H
Part No. 88SD22 (Fits into DTE-20 Enclosure) . . . . $69.95 each

POWER SUPPLY — 5VDC @ 1 AMP REGULATED — Transaction Tech
Output +5VDC @ 1 amp (also +30VDC) reg. Input 115VAC 60Hz. Two-tone (black/beige) self-enclosed case. 6 ft., 3 cond. black power cord. Size: 6⅝"W x 7"D x 2¼"H. Wt. 3 lbs.
Part No. PS51194 . . . . $19.95 each

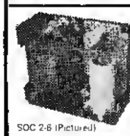
POWER SUPPLY — 5VDC @ 1 AMP REGULATED — B Industries
Output +5VDC @ 1 amp, +36-42VDC adj. 400mA or less, 30VAC (isol.) @ 1.5 amp. Input 115VAC 60Hz. Circ. brkr. re-set button. Blk. self-encl. case w/4 rubber feet. 6 ft., 3 cond. blk. pow. cord. On/off switch. 6¾"W x 7¾"D x 3-7/8"H - wt. 7 lbs.
Part No. PS4070 . . . . $24.95 each

POWER SUPPLY — 5VDC @ 3 AMP REGULATED — Deltron
Input 115VAC, 47-440Hz. Output: 5VDC Adjustable @ 3 amp, 6VDC @ 2.5 amp. Adjustable current limit, Ripple & Noise 1MV rms, 5MV p-p — 2 mounting surfaces. UL recognized. Size: 4"W x 4½"L x 2-7/16"H - wt. 2 lbs.
Part No. OPS-1 . . . . $29.95 each

POWER SUPPLY — 5VDC @ 7.5 AMP, 12VDC @ 1.5 AMP SWITCHING
Input: 115VAC, 50-60Hz @ 3 amp/230VAC, 50Hz @ 1.6 amp. Fan volt./power supply select switches (115/230VAC). Output: 5VDC @ 7.6 amp, 12VDC @ 1.6 amp. 6 ft. blk. pow. cord. 11½"W x 13¼"D x 3¾"H. Wt. 6 lbs.
Part No. PS94V0 . . . . $49.95 each

MULTI-VOLTAGE POWER SUPPLY — +5,+12,−12VDC REG.
Input: 105-125VAC, 47-63Hz/205-250VAC, 47-63Hz. Output: +5VDC @ 2 amps Adj., 5VDC @ 50mA Fixed, +12VDC @ 1 amp Adj., −12V @ 2 amp adj. Overvoltage protection. Size: 12½"L x 4-7/8"W x 3¾"D.
Part No. RA0250 . . . . $39.95 each

SORENSEN Regulated Power Supplies



SOC 2-6 (Pictured)

Sorensen's open construction (SOC) power supplies are series-regulated solid-state systems, designed to provide reg. DC voltages at 6 levels (2-28 v/range). These units are open-framed on sturdy black anodized aluminum for excellent mounting.

FEATURES: 115/208/230VAC Input @ 50-63Hz. Low Ripple 1.5mVrms, 5mV P-P maximum. Adjustable current limit. Voltage adjustment control. All schematics and specifications supplied with unit. Series A,B,C,E have three mounting surfaces (Series F, bottom mounting only).

Part No	Series	Output Voltage Adjustment Range min	max.	Output Current amps (Abc) @40°C	@50°C	@60°C	Size (inches)	Weight	Price
SOC 2-6	B	1.9	2.1	6.0	4.9	3.8	5.62 x 4.88 x 2.50	4.3 lbs.	$19.95
SOC 2-25	F	1.9	2.1	25.0	21.5	17.5	16.00 x 4.88 x 4.88	16 lbs.	29.95
SOC 5-18	E	4.75	5.25	18.0	15.0	12.0	14.00 x 4.88 x 2.25	12 lbs.	28.95
SOC 5-25	F	4.25	5.25	25.0	21.5	17.5	16.00 x 4.88 x 4.88	16 lbs.	49.95
SOC 12-11	E	11.4	12.6	11.0	9.2	6.8	14.00 x 4.88 x 1.62	12 lbs.	44.95
SOC 12-15	F	11.4	12.6	15.0	12.25	9.5	16.00 x 4.88 x 4.88	16 lbs.	49.95
SOC 15-5	C	14.25	15.75	5.0	4.2	3.5	7.00 x 4.88 x 3.37	6.6 lbs.	38.95
SOC 15-9.5	E	14.25	15.75	9.5	7.6	5.6	14.00 x 4.88 x 1.62	12 lbs.	44.95
SOC 15-13	F	14.25	15.25	13.0	10.5	8.0	16.00 x 4.88 x 4.88	16 lbs.	49.95
SOC 28-0.8	A	26.6	29.4	0.8	.64	.45	4.00 x 4.88 x 1.62	2 lbs.	24.95

Powertec Sub-Modular DC Power Supplies

22D300 (Pictured)

SM Series power supplies include rectifying, filtering, regulating, overload and overvoltage protection functions. You need only connect the sub-module to the appropriate secondary transformer tap and bolt the unit to a heatsink.

REGULATION: LINE: .10% for a change from -10% to +10% input voltage. LOAD: .15% for a 0-100% load change (units below 5V output maintain 5V regulation). OUTPUT RIPPLE: 1mV RMS, 3mV P-P typical, 5mV P-P maximum. INPUT CHARACTERISTICS: Requires low-level AC input. Derate output current 15% for operations at 50Hz.

Part Number	-100 *4.75V to 7.0V	-200 *7.0V to 10.5V	-300 *10.5V to 15.25V	-500 *22.0V to 30.0V	Power Transformer Requirements (Transformers Not Included) Primary	Secondary	Size (inches)	Wt	Price
22A-300			0.23A		115–120VAC	12VAC .5A no CT	2.50 x 3.00 x .98	2 oz.	$14.95
22B-200		2.2A			115–120VAC	22VAC 3A w/CT	3.00 x 5.75 x 1.18	8 oz.	14.95
22B-300			1.7A		115–120VAC	28VAC 2.5A w/CT	2.75 x 5.75 x 1.18	8 oz.	19.95
22C-100	6.0A				115–120VAC	16VAC 6A w/CT	2.80 x 7.50 x 1.18	8 oz.	24.95
22C-500				2.5A	115–120VAC	48VAC 3A w/CT	2.80 x 7.50 x 1.18	8 oz.	20.95
22D-300			6.8A		115–120VAC	28VAC 10A w/CT	3.00 x 7.00 x 3.30	2 lbs.	24.95
22E-100	18.00A				115–120VAC	16VAC 24A w/CT	2.00 x 7.00 x 3.30	2 lbs.	29.95

*Voltage Adjustment Range — Current ratings apply over entire voltage range

★ SHIPMENT IN 24 HOURS ★
MONDAY THRU FRIDAY 7:00AM to 5:00PM (PST) NEW HOURS
Call: (415) 592-8097

JUMPER AND CABLE ASSEMBLIES

Custom Jumpers! — Custom Cables!

STANDARD DIP JUMPERS

All jumpers use low profile dip plugs with heavy duty pins for repeated disconnect applications

JAMECO Part No	AP Cross Reference	No Pins	Description	Wire Length	Price
DJ14-1	924102-12	14	single end	12"	$1.29
DJ14-2	924102-24	14	single end	24"	2.00
DJ14-3	924102-36	14	single end	36"	2.39
DJ14-1-14	924106-12	14	double end	12"	3.29
DJ14-2-14	924106-24	14	double end	24"	3.49
DJ14-3-14	924106-36	14	double end	36"	3.29
DJ16-1	924112-12	16	single end	12"	1.85
DJ16-2	924112-24	16	single end	24"	2.19
DJ16-3	924112-36	16	single end	36"	2.59
DJ16-1-16	924116-12	16	double end	12"	3.35
DJ16-2-16	924116-24	16	double end	24"	3.69
DJ16-3-16	924116-36	16	double end	36"	4.05
DJ24-1	924122-12	24	single end	12"	2.00
DJ24-2	924122-24	24	single end	24"	3.39
DJ24-3	924122-36	24	single end	36"	3.96
DJ24-1-24	924126-12	24	double end	12"	4.20
DJ24-2-24	924126-24	24	double end	24"	5.79
DJ24-3-24	924126-36	24	double end	36"	5.49
DJ40-1	924132-12	40	single end	12"	5.80
DJ40-2	924132-24	40	single end	24"	6.29
DJ40-3	924132-36	40	single end	36"	7.69
DJ40-1-40	924136-12	40	double end	12"	10.05
DJ40-2-40	924136-24	40	double end	24"	11.09
DJ40-3-40	924136-36	40	double end	36"	12.29

STANDARD DB25 SERIES CABLES

Now you can order DB25 P or S connectors with the cable necessary to fit your application. Choose from our standard flat cable in 4 foot lengths. **Call today.**

STANDARD CABLES

Part No.	Cable Length	Connectors	Price
DB25P-4	4 feet	1 DB25P	$ 9.95 ea.
DB25S-4	4 feet	1 DB25S	10.95 ea.
DB25P-4-P	4 feet	2 DB25P	16.95 ea.
DB25P-4-S	4 feet	1 DB25P/1 DB25S	17.95 ea.
DB25S-4-S	4 feet	2 DB25S	18.95 ea.

$10.00 Minimum Order — U.S. Funds Only
California Residents Add 6½% Sales Tax
Postage — Add 5% plus $1.50 Insurance
Send S.A.S.E. for Monthly Sales Flyer!

Spec Sheets — 30¢ each
Send $1.00 Postage for your
FREE 1983 JAMECO CATALOG
Prices Subject to Change

Call for Quantity Discounts

Mail Order Electronics · Worldwide
Jameco ELECTRONICS

NEW! Telex Number Telex 176043

1355 SHOREWAY ROAD, BELMONT, CA 94002
11/82 *PHONE ORDERS WELCOME — (415) 592-8097* 534

5¼" Mini-Floppy Disc Drive

FOR TRS-80 MODEL I (Industry Standard)

Pertec Computer Corp.

Features single or double density. Recording mode FM single, MFM double density. Power: +12VDC (±0.6V) 1.6 amps max., +5VDC (±0.25V) 0.8 amps max. Unit as pictured at left (does not incl. case, power supply or cables). 30-page data book included. Weighs 3½ pounds. Size: 5¾"W x 8"D x 3¼"H

Limited Quantity!

Part No. — Price
FD200 . . . . $179.95
Single-sided, 40 tracks, 250K bytes capacity
FD250 . . . . $199.95
Double-sided, 35 tracks, 438K bytes capacity

EXPAND YOUR TRS-80

to 16K, 32K, or 48K

**Model 1 = From 4K to 16K Requires (1) One Kit
Model 3 = From 4K to 48K Requires (3) Three Kits
Color = From 4K to 16K Requires (1) One Kit

**Model 1 equipped with Expansion Board up to 48K Two Kits Required — One Kit Required for each 16K of Expansion —

TRS-80 16K Conversion Kit

Kit comes complete with 8 each MM5290 (UPD416/4116) 16K Dynamic RAM (*ns) and documentation for conversion.

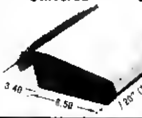
TRS-16K2 *150ns $14.95
TRS-16K3 *200ns $12.95
TRS-16K4 *250ns $10.95

TRS-80 Color 32K or 64K Conversion Kit

Kit comes complete with 8 ea. 4164-2 (200ns) 64K Dyn. RAMs & conversion documentation. Converts TRS-80 color computers from 4K-32K Memory or 16K-64K Memory

TRS-64K2 (200ns) $54.95

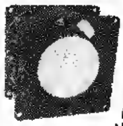
Universal Computer Keyboard Enclosure



"DTE" Blank Desk-Top Enclosures are designed for easy modification. High strength epoxy molded end pieces in mocha brown finish. Sloping rear/bottom panel for service/component access. Top/bott. panels .080" thick alum. alodine type 1200 finish (gold tint color) for best paint adhesion after modification. Vented top & bottom panels for cooling efficiency.

DTE-20 Panel width 20" . . . $39.95

Sprite-style Fan



- 36cfm free air delivery
- 3.125" sq. x 1.665" depth
- 10 yrs. cont. duty at 20°C
- 115V 50/60Hz
- For Apple users

Metal Frame

PWS2107U Cleaned & tested (used) . . $ 9.95 ea.
*PWS2107F New . . . $14.95 ea.

Muffin-style Fan

- 105cfm free air delivery
- 4.68" sq. x 1.50" depth.
- 10 yrs. cont. duty at 20°C
- Impedance protected, ambients to 70°C
- 115V 50/60Hz 14W Wt. 17 oz.

Metal Frame

MU2A1-U Cleaned & tested (used) . . . . . $9.95 ea.
*MU2A-1N New . . . . . $14.95 ea.

JOYSTICKS

JS-5K (Pictured)

JS-5K	5K Linear Taper Pots	$5.25
JS-100K	100K Linear Taper Pots	$4.95
JS-150K	150K Linear Taper Pots	$4.75
JVC-40	40K (2) Video Controller in Case	$4.95

UV-EPROM Eraser

8 Chips — 51 Minutes

1 Chip — 37 Minutes

Erases 2708, 2716, 2732, 2764, 2516, 2532, 2564. Erases up to 8 chips within 51 minutes (1 chip in 37 minutes). Maintains constant exposure distance of one inch. Special conductive foam liner eliminates static build-up. Built-in safety lock to prevent UV exposure. Compact — only 9.00" x 3.70" x 2.60". Complete with holding tray for 8 chips.

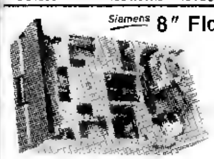
UVS-11EL Replacement Bulb $16.95
DE-4 UV-EPROM Eraser $79.95

Wall Transformers AC and DC Types

AC250 (Pictured)

Part No.	Input	Output	Price
AC 250 (above)	117V/60Hz	12VAC 250mA	$3.95
AC 500	117V/60Hz	12VAC 500mA	$4.95
AC1000	117V/60Hz	12VAC 1 amp	$5.95
AC1700	117V/60Hz	9VAC 1.7 amp	$3.95
AC9004	117V/60Hz	9.2VAC 2.5 amp	$2.95 or 2/$4.95
DC 800	120V/60Hz	8VDC 400mA	$1.95 or 2/$2.95
DC6912	120V/60Hz	6,9,12VDC 300mA	$9.95
DV5490	117V/60Hz	9.5VDC 275mA	$2.49 or 2/$3.95
DC900	120V/60Hz	9VDC 500mA	$3.95
DC1200	120V/60Hz	12VDC 300mA	$2.95 or 2/$4.95

Siemens 8" Floppy Disk Drive



- Single-Sided
- 77 Tracks
- 400/800K Bytes Capacity
- Industry Standard

The FDD100-8 8" Floppy Disk Drive (Industry Standard) features single or double density. Recording mode: FM single, MFM double density. Transfer rate: 250K bits/sec. single density, 500K bits/sec. double density. The FDD100-8 is designed to work with the single-sided soft sectored IBM Diskette 1, or equivalent disk cartridge. A hard-sectored option is available. Power: 115/230VAC @ 50-60Hz, +24VDC @ 1.7 amps max., +5VDC @ 1.2 amps max. Unit as pictured above (does not include case, power supply, or cables). Size: 9.55"W x 14"L x 4.5"H Weighs 12 lbs

Part No. — Price
FDD100-8 Buy 1 for . . . . $299.95 each
FDD100-8 Buy 2 for . . . . $279.95 each
FDD100-8 Buy 10 for . . . . $249.95 each

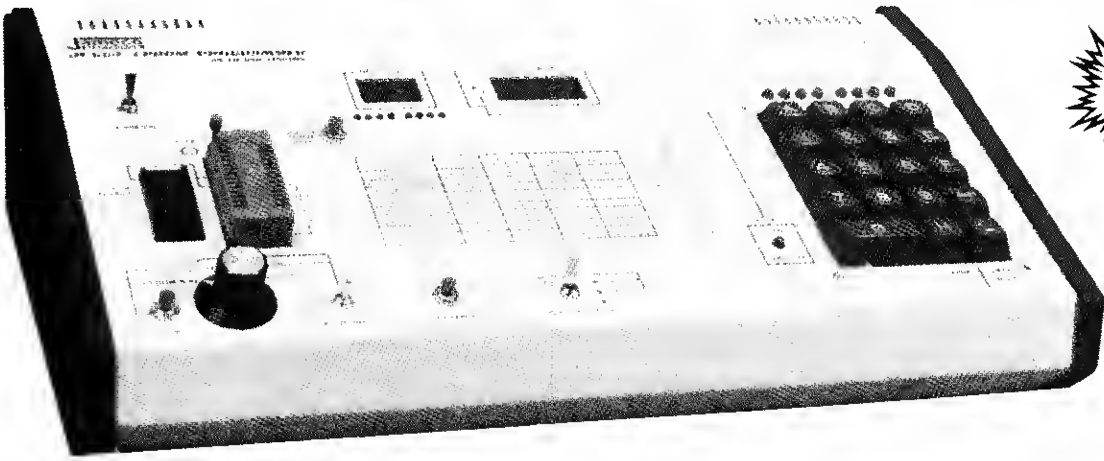
Mail Order Electronics · Worldwide
Jameco ELECTRONICS

2708,2716,2732 AND 2764 EPROM PROGRAMMER

JE664 EPROM PROGRAMMER

8K TO 64K EPROMS — 24 AND 28 PIN PACKAGES

COMPLETELY SELF-CONTAINED — REQUIRES NO ADDITIONAL SYSTEMS FOR OPERATION



• Programs and validates EPROMs • Checks for properly erased EPROMs • Emulates PROMs or EPROMs • RS232C Computer Interface for editing and program loading • Loads data into RAM by keyboard • Changes data in RAM by keyboard • Loads RAM from an EPROM • Compares EPROMs for content differences • Copies EPROMs • Power Input: 115VAC, 60Hz, less than 10W power consumption • Enclosure: Color-coordinated, light tan panels with molded end pieces in mocha brown • Size: 15-5/8"L x 8¾"D x 3½"H • Weight: 5¾ lbs.

The JE664 EPROM Programmer emulates and programs various 8-Bit Word EPROMs from 8K to 64K-Bit memory capacity. Data can be entered into the JE664's internal 8K x 8-Bit RAM in three ways: (1) from a ROM or EPROM; (2) from an external computer via the optional JE665 RS232C BUS; (3) from its panel keyboard. The JE664's RAMs may be accessed for emulation purposes from the panel's test socket to an external microprocessor. In programming and emulation, the JE664 allows for examination, change and validation of program content. The JE664's RAMs can be programmed quickly to all "1"s (or any value), allowing unused addresses in the EPROM to be programmed later without necessity of "UV" erasing. The JE664 displays DATA and ADDRESS in convenient hexadecimal (alphanumeric) format. A "DISPLAY EPROM DATA" button changes the DATA readout from RAM word to EPROM word and is displayed in both hexadecimal and binary code. The front panel features a convenient operating guide. The JE664 Programmer includes one JM16A Jumper Module (as listed below).

JE664-A **EPROM Programmer** — Assembled & Tested (Includes JM16A Module)............ **$995.00**

JE665 — RS232C INTERFACE OPTION — The JE665 RS232C Interface Option implements computer access to the JE664's RAM. A sample of software written in BASIC is provided for the TRS-80® Model I, Level II Computer. Baud rate: 9600. Word Length: 8 Bits — Odd Parity. Stop Bits: 2. This option may be readily adapted to other computers.

JE664-ARS **EPROM Programmer with JE665 Option**............................ **$1195.00**
Assembled and Tested (Includes JM16A Module)

EPROM JUMPER MODULES — The JE664's JUMPER MODULE (Personality Module) is a plug-in Module that pre-sets the JE664 for the proper programming pulses to the EPROM and configures the EPROM socket connections for that particular EPROM.

JE664 EPROM JUMPER MODULE NO.	EPROM	EPROM MANUFACTURER	PRICE
JM08A	2708	AMD, Motorola, National, Intel, TI	$14.95
JM16A	2716, TMS2516(TI)	Intel, Motorola, National, NEC, TI	$14.95
JM16B	TMS2716 (3 Voltages)	Motorola, TI	$14.95
JM32A	TMS2532	Motorola, TI	$14.95
JM32B	2732	AMD, Fujitsu, NEC, Hitachi, Intel	$14.95
JM64A	MCM68764, MCM68L764	Motorola	$14.95
JM64B	2764	Intel	$14.95
JM64C	TMS2564	TI	$14.95

Mail Order Electronics · Worldwide
Jameco ELECTRONICS

1355 Shoreway Road, Belmont, CA 94002 ✓534 FOR FAST DELIVERY CALL (415) 592-8097

Super Spooler

by Ron Balewski

Don't let your printer tie up your computer during long tasks. Use this spooler, instead—it doesn't even require more hardware.

Do you have a *slow* printer that bogs down your TRS-80? How would you like to run another program, type in another program, or maybe even play a game of Star Trek while your printer grinds out your mailing labels? Not possible without some sort of extra hardware, right? Wrong. A print spooler will do everything I just described.

What Is a Print Spooler?

A spooler is a program that compensates for the slowness of a printer by accepting printed data as fast as the Basic interpreter can send it, and storing it in a buffer for later release to the printer at a speed slow enough for the printer to handle.

There are two types of spoolers, RAM based and disk based. Since I don't have a disk system, I use a RAM-based spooler. I reserve the top 32K (or however much I can do without) of memory for the spooler print buffer. This lets me store about 32,700 characters in the buffer. On my Centronics 737, which prints at 50 characters per second, this amounts to about 10 minutes printing time.

Use a disk-based spooler if you have a disk system. With this kind of spooler, printed data is not stored in RAM (which is usually in short supply), but is spooled out to a disk. Whenever the printer requests more data, it's pulled back off the disk and sent to the printer. A typical 35-track disk should hold about 89,000 bytes of data.

On my 50-character-per-second 737, this would amount to almost a half hour printing time, obviously better than the 10 minutes offered by the RAM spooler.

How It Works

The print-spooler program takes advantage of the interrupt generated by the expansion interface so the computer appears to be doing two things at once: controlling the printer and running another program.

The first part of the program, up to line 650, is a new line printer-driver routine. This is patched into the line-printer DCB (data control block), bypassing Radio Shack's driver. Lines 230–270 make sure there's room in the buffer for another character. Lines 280–360 put the new character into the buffer and adjust the input pointer. Lines 470–650 contain a subroutine that is called when the print buffer is full. "Full" is printed on the screen by lines 480–510. Lines 530–570 then wait until the buffer empties a bit. The "Full" is then erased and control returns to the main driver.

The second main part of the program, lines 770–1240, is the interrupt service routine. This section is called 40 times per second, regardless of what the TRS-80 is doing. During each call, the routine first checks if there's any data in the buffer. If not, the interrupt ends. If there is data in the buffer, the routine checks if the printer can accept some. If not, the interrupt ends.

Assuming there is data to be printed and the printer is ready to accept data, the interrupt service routine sends the next byte of data to the printer, delays for an instant, and loops back to perform the above two checks again.

Note that the delay loop in lines 990–1000 may have to be adjusted or possibly removed entirely, depending on your printer. My 737 has an 80-character print buffer built in and only prints after either 80 characters or certain control codes are received. Therefore, my interrupt service routine is designed to pass data to the printer in blocks instead of passing one character per interrupt (since the maximum speed at one character per interrupt is only 40 characters per second). When-

> *The Key Box*
>
> **Model I**
> **32K RAM**
> **Cassette Basic, Assembly**

ever a character is passed to the 737, though, it takes the printer a moment to decide whether or not it can accept another character before it must start printing. The 30 in line 990 reflects this time delay.

If your printer "thinks" faster, you can lower this number. You'll know if that number is too low because your printer will print slower than its maximum speed due to the fact that it's only getting 40 characters per second and not its maximum amount. If your printer has no input buffer, you can delete the delay loop entirely since you'll return from the interrupt after each character printed.

If your printer has no print buffer, the maximum speed it prints at under this spooling program is 40 characters per second, despite its maximum speed rating.

"The initialization section starts the ball rolling by enabling the interrupts and then executing a Clear."

The next three program sections, when called, kill all the data in the print buffer, enable interrupts to start the printer, and disable interrupts to stop printing, respectively. All three of these sections are patched into DOS reserved words and are explained more fully in the operating instructions.

The final section, lines 1610–1640, is the initialization section. It is run only once, immediately after you load the program. It starts the ball rolling by enabling the interrupts and then executing a Clear to fix the pointers messed up while loading.

The rest of the program just sets memory size for you and patches in the new printer driver, the interrupt service routine, and the spooler commands.

How to Use It

Now that you know the basics of how this spooler works, let me tell you about its special features and how to use them. The spooler responds to three DOS key words—Open, Close, and Kill. Open enables the interrupts and starts the printer printing. Close disables the interrupts and stops the printer. Of course, you'll still be able to do LPRINTs, even though the printer

Program Listing

```
                 00100 ;          ***********************
                 00110 ;          * RAM SPOOLER PROGRAM *
                 00120 ;          ***********************
                 00130 ;             BY RON BALEWSKI
                 00140 ;              JULY 17, 1981
                 00150 ;
                 00160 ; LINE PRINTER DRIVER ROUTINE - STORES ALL LPRINTED
                 00170 ;       DATA IN THE SPOOLING BUFFER.
3C00             00180 VIDEO    EQU     3C00H
8000             00190          ORG     8000H              ;TOP 32K
8000 C5          00200 LPRDVR   PUSH    BC
8001 D5          00210          PUSH    DE
8002 E5          00220          PUSH    HL
8003 ED5BB480    00230          LD      DE,(INPPTR)        ;BE SURE ROOM IN BUFFER
8007 2AB680      00240          LD      HL,(OUTPTR)
800A 13          00250          INC     DE
800B DF          00260          RST     18H
800C CC2380      00270          CALL    Z,BUFFUL           ;IF NO ROOM, ? MSG & WAIT
800F 79          00280          LD      A,C
8010 2AB480      00290          LD      HL,(INPPTR)        ;PUT PRINTED CHAR IN BFR
8013 77          00300          LD      (HL),A
8014 23          00310          INC     HL                 ;INCREMENT INPUT POINTER
8015 7C          00320          LD      A,H
8016 B5          00330          OR      L
8017 2003        00340          JR      NZ,LPRXIT
8019 21B880      00350          LD      HL,BUFFER
801C 22B480      00360 LPRXIT   LD      (INPPTR),HL
801F E1          00370          POP     HL
8020 D1          00380          POP     DE
8021 C1          00390          POP     BC
8022 C9          00400          RET
                 00410 ;
                 00420 ;
                 00430 ; BUFFUL - THIS SUBROUTINE PRINTS FULL IN THE LOWER LEFT
                 00440 ;          CORNER OF THE VIDEO SCREEN AND WAITS UNTIL THE
                 00450 ;          BUFFER EMPTIES SOME.  IT THEN ERASES FULL AND
                 00460 ;          RETURNS.
8023 C5          00470 BUFFUL   PUSH    BC
8024 010400      00480          LD      BC,4
8027 11F73F      00490          LD      DE,VIDEO+1015
802A 214880      00500          LD      HL,FULMSG
802D EDB0        00510          LDIR
802F FB          00520          EI
8030 ED5BB480    00530 STLFUL   LD      DE,(INPPTR)
8034 2AB680      00540          LD      HL,(OUTPTR)
8037 13          00550          INC     DE
8038 DF          00560          RST     18H
8039 28F5        00570          JR      Z,STLFUL
803B 010400      00580          LD      BC,4
803E 11F73F      00590          LD      DE,VIDEO+1015
8041 214C80      00600          LD      HL,BLANKS
8044 EDB0        00610          LDIR
8046 C1          00620          POP     BC
8047 C9          00630          RET
8048 46          00640 FULMSG   DEFM    'FULL'
804C 20          00650 BLANKS   DEFM    '    '
                 00660 ;
                 00670 ;
                 00680 ; INTERRUPT SERVICE ROUTINE - THIS ROUTINE IS ACCESSED
                 00690 ;      40 TIMES A SECOND.  IT CHECKS IF THERE IS DATA
                 00700 ;      IN THE BUFFER AND IF THE PRINTER WILL ACCEPT SOME.
                 00710 ;      IF SO, IT OUTPUTS DATA UNTIL THE PRINTER BUFFER
                 00720 ;      IS FULL (PRINTER REFUSES TO ACCEPT MORE DATA),
                 00730 ;      AND THEN RETURNS TO DO BASIC UNTIL IT CAN PRINT
                 00740 ;      AGAIN.
                 00750 ;
                 00760 ;
8050 F3          00770 INTRTN   DI
8051 F5          00780          PUSH    AF
8052 C5          00790          PUSH    BC
8053 D5          00800          PUSH    DE
8054 E5          00810          PUSH    HL
8055 ED5BB480    00820 NXTOUT   LD      DE,(INPPTR)        ;CHECK IF DATA IN BUFFER
8059 2AB680      00830          LD      HL,(OUTPTR)
805C DF          00840          RST     18H
805D 2824        00850          JR      Z,NOPRNT
805F CDD105      00860          CALL    05D1H              ;CHECK IF PRINTER READY
8062 201F        00870          JR      NZ,NOPRNT
8064 2AB680      00880          LD      HL,(OUTPTR)        ;GET NEXT LETTER
8067 7E          00890          LD      A,(HL)
8068 F5          00900          PUSH    AF
8069 23          00930          INC     HL                 ;INC OUTPUT POINTER
806A 7C          00940          LD      A,H
806B B5          00950          OR      L
806C 2003        00960          JR      NZ,NORES
806E 21B880      00970          LD      HL,BUFFER
8071 22B680      00980 NORES    LD      (OUTPTR),HL
8074 F1          00981          POP     AF
8075 B7          00982          OR      A                  ;LETTER 0?
8076 281A        00983          JR      Z,PRTOFF           ;IF SO, THEN OFF PRINTER
8078 32E837      00984          LD      (37E8H),A          ;PRINT THE LETTER
807B 011E00      00990          LD      BC,30              ;LET PRINTER ACCEPT DATA
807E CD6000      01000          CALL    0060H
8081 18D2        01010          JR      NXTOUT
8083 2AEC37      01020 NOPRNT   LD      HL,(37ECH)
8086 2AE037      01030          LD      HL,(37E0H)
8089 2AE037      01040          LD      HL,(37E0H)
```

Listing continues

isn't going, since all printed data just goes into a buffer. Kill kills all data in the print buffer. This is a handy feature in case you accidentally do an LLIST. It won't take long to dump a lot of garbage to the print buffer. Kill gives you a way to throw out the garbage without resetting the system.

Another nice feature of the spooler is a print-hold feature. Since I frequently use single-sheet paper with my printer, I include in many of my programs an automatic stop after so many lines printed so I can put a new sheet of paper into the printer. Such a pause works fine when Basic is running the printer directly. Whenever Basic stops, the printer stops. However, things don't work quite so smoothly when a spooler is put between Basic and the printer.

In the blink of an eye, the computer was telling me to change the paper and press enter, but the printer was just getting started on that page! If I waited until the page was printed, changed the paper, and then pressed enter, I defeated the purpose of the spooler by once again making the computer wait for the printer. On the other hand, if I pressed enter and let the computer continue to pour data into the buffer, all

Listing continued

```
808C E1                POP     HL
808D D1                POP     DE
808E C1                POP     BC
808F F1                POP     AF
8090 FB                EI
8091 C9                RET
                ;
                ; WHEN A ZERO IS ENCOUNTERED IN THE PRINT BUFFER, CONTROL
                ;       BRANCHES HERE TO DISABLE THE INTERRUPTS AND
                ;       THEREFORE STOP THE PRINTER.  THIS ALLOWS YOU TO
                ;       PUT PAUSES IN THE PRINTING FOR FORMS CHANGING.
                ;       TYPING OPEN WILL RE-START THE PRINTING.
8092 2AEC37     PRTOFF LD      HL,(37ECH)
8095 2AE037            LD      HL,(37E0H)
8098 2AE037            LD      HL,(37E0H)
809B E1                POP     HL
809C D1                POP     DE
809D C1                POP     BC
809E F1                POP     AF
809F C9                RET
                ;
                ;
                ;
                ; KILL - THE KILL COMMAND WILL KILL ALL DATA IN THE PRINT
                ;        QUEUE. THIS IS PROVIDED IN CASE A LOT OF USELESS
                ;        DATA IS ACCIDENTALLY DUMPED TO THE PRINTER.
80A0 21B880     KILL   LD      HL,BUFFER       ;CLEAR THE PRINT BUFFER
80A3 22B480            LD      (INPPTR),HL
80A6 22B680            LD      (OUTPTR),HL
80A9 C3191A            JP      1A19H
                ;
                ;
                ;
                ; OPEN - THE OPEN COMMAND WILL TURN ON THE INTERRUPT AND
                ;        THEREFORE TURN ON PRINTING.
80AC FB         OPEN   EI                      ;ENABLE INTERRUPTS
80AD C3191A            JP      1A19H
                ;
                ;
                ;
                ; CLOSE - THE CLOSE COMMAND WILL DISABLE THE INTERRUPTS
                ;         AND THEREFORE STOP THE PRINTING.  THIS IS
                ;         USEFUL BECAUSE YOU CAN'T READ OR WRITE A TAPE
                ;         WHILE THE INTERRUPT IS ENABLED.  TO RE-START
                ;         PRINTING, TYPE OPEN.
80B0 F3         CLOSE  DI                      ;DISABLE INTERRUPTS
80B1 C3191A            JP      1A19H
80B4 B880       INPPTR DEFW    BUFFER
80B6 B880       OUTPTR DEFW    BUFFER
80B8 00         BUFFER DEFB    0
                ;
                ;
                ;
                ; INITIALIZATION SECTION - AFTER INITIALIZATION, THIS
                ;       CODE WILL BE OVERWRITTEN AS PART OF THE PRINT
                ;       BUFFER.
80B9 FB         INIT   EI
80BA 21191A            LD      HL,1A19H
80BD E5                PUSH    HL              ;SET A RETURN ADDRESS
80BE C37A1E            JP      1E7AH           ;DO A CLEAR
                ;
                ;
                ;
                ; SET PRINTER DCB ADDRESS
4026                   ORG     4026H
4026 0080              DEFW    LPRDVR
                ;
                ;
                ;
                ; SET INTERRUPT ADDRESS
4012                   ORG     4012H
4012 C35080            JP      INTRTN
                ;
                ;
                ;
                ; RESET TOP-OF-MEMORY POINTER
40B1                   ORG     40B1H
40B1 FF7F              DEFW    LPRDVR-1
                ;
                ;
                ;
                ; RESET MEMORY SIZE PTR TO CLEAR 50
40A0                   ORG     40A0H
40A0 CD7F              DEFW    LPRDVR-51
                ;
                ;
                ;
                ; SET KILL DOS VECTOR ADDRESS
4191                   ORG     4191H
4191 C3A080            JP      KILL
                ;
                ;
                ;
                ; SET OPEN DOS VECTOR ADDRESS
4179                   ORG     4179H
4179 C3AC80            JP      OPEN
                ;
                ;
                ;
                ; SET CLOSE DOS VECTOR ADDRESS
4185                   ORG     4185H
4185 C3B080            JP      CLOSE
80B9                   END     INIT
00000 TOTAL ERRORS
```

the output ran together without the planned pause.

The addition of the above-mentioned print-hold feature solved the problem. To turn off (Close) the printing at a given point, just LPRINT CHR$(0). The printer prints up to the zero in the buffer and stops until you issue an Open command. You can include as many zeros in the print buffer as you want. The printer stops whenever it hits one.

If the buffer becomes full when the interrupt is off (closed), the interrupt is automatically enabled (opened) to keep the system from eternally waiting for the buffer to empty, even though the printer is closed.

Now for a bit of advice: Be sure to Close the printer before you read or write a cassette tape. If you try to do tape I/O with the interrupt enabled, the time spent in the interrupt service routine wreaks havoc with the tape timing. Don't worry about freely using Opens and Closes since no data is lost during their use.

You can change the size of the buffer by changing the ORG in line 190. All memory from the ORG value through FFFH is assumed to be available to the spooler. Using an ORG value of 8000H gives you 32K to be used by the spooler and 16K to be used by your program. An ORG value of C000H gives you 32K of memory for your program while leaving only 16K for the spooler. I keep an object file assembled with each of the two ORG values handy and reserve 32K for the spooler whenever possible. But if I can't afford 32K, I just load the copy that reserves 16K.

For Use With 32K

If you have a 32K system, you can get this spooler to run just by changing lines 330 and 950 to CP 0C0H. All this does is check to see if you went above BFFFH, the end of a 32K machine instead of FFFFH.

Once you use this spooler for a while, you'll never run your printer again without it.

Now for a challenge to all you 5¼-inch disk jockeys out there: How about writing a spooling program that uses a disk for temporary storage? That way, I won't have to write one when I get a disk system. ■

Ron Balewski (412 E. Ridge St., Nanticoke, PA 18634) is a freelance programmer. His interests include home video, community theater, and electronics.

See List of Advertisers on Page 563

Election

by Robert Jacobs

Election teaches students about campaigning by introducing them to several facets of elections such as budgets and stands on issues.

Election simulates a Congressional District campaign and election. It incorporates the strategies of recent Congressional campaigns as well as those of other medium-sized campaigns. This simulation is valuable for teaching students about the political choices and strategies available to candidates for public office in the early 1980s.

You can alter the program to account for local political conditions, changing issues, or fluctuating interests in American politics. The calculations of the control functions have been kept separate from one another to allow you to alter the weights given the variables and to change the assumptions upon which the program is based. Although the assumptions are not extremely controversial, their applicability will vary for different sections of the country and from one election to the next. You, as the instructor, should familiarize yourself with the assumptions.

The electorate is established as 52 percent Democrat and 48 percent Republican. This roughly reflects the total national Congressional vote in the 1980 election.

Regardless of other factors, the incumbent candidate receives .75 percent of the ultimate turnout for each week of the campaign. This reflects the American political scene, although situational factors may alter this percentage.

The amount of funds a candidate may raise has a predetermined ceiling. Students can discover this limit, as well as the proximate return for fund-raising efforts, by trial and error.

The voter turnout is set low, but it can be increased by taking controversial stances on issues. However, controversial positions can also be detrimental to the student's candidacy, so the student must find the balancing point between the two extremes.

The program minimizes the impact of candidates' positions on issues. In some American elections, the issues dominate, but in most elections the candidates' positions become blurred and are almost indistinguishable from one another. The power of the positions can be changed, but I don't recommend this because the students' participation will become merely a hunt for "buried treasure," with the treasure being the "correct" stand on an issue.

Changing positions too frequently results in a penalty.

The effect of news headlines on the election is restricted. However, the weight and directional bias of the news stories may be changed by the instructor.

The student must allocate funds for several forms of mass communication to avoid erosion of support. The program ascribes heavier weight to television expenditures, since television is our dominant medium. The order of efficacy is: television, radio, newspaper advertising, and campaign souvenirs. The latter two do little to promote the candidacy, but they prevent erosion of support. This order of importance can be altered.

Using the Program

Election is written in Basic, and it uses most of the 16K. It is CLOADable on both Model I and III. Single characters, both alphabetic and numeric, are entered by single key strokes through the INKEY$ function; when entering dollar amounts, press enter.

After listing the instructions, the

The Key Box

Model I or III
16K, 32K RAM
Cassette or Disk Basic

program asks the student to choose a political party and decide whether or not he wants to be the incumbent. Then, the student must allocate campaign funds among the various media. If the student errs or spends more than is allocated, the program loops back so he can redo the budget.

The student then selects political positions on five issues. The range of choices span the range of "respectable" opinions as reported by American opinion polls. After these positions are entered, the first of several staff reports is offered. This lists the results of the week's polls, and it reports on the budget.

The campaign runs for 10 weeks. After each week, news headlines appear, and the student is asked whether or not he wishes to revise any of the positions selected. A new budget must also be prepared. The staff report gives the student a running report of the campaign's progress.

At the end of the 10 weeks, the results of the election are announced, along with a summary of the monies expended and the number of position changes.

Instructors who think the student will gain access to the program listing to either change the variables or discover how to win can disable the break key by altering line 10 as follows:

```
10 POKE 16396, 175: POKE 16397, 201
```

To restore the key, POKE 16396, 201 from the command level or reset the computer. With disk systems on the Model I, different locations must be accessed. (See Table 1.)

Most Model III operating systems have a built-in break disable.

Classroom Use

For the most effective classroom use of Election, the students should pay attention to more than just winning the election; they can learn from the program only if it is approached systematically.

Several teaching strategies are available. The most useful of these is to divide the class into groups of participants, and, for each group, to hold one or more of the program variables constant. For example, one group of students might be required to be the incumbent while the other might be the challenger; one group might be Democrats and the other Republicans.

A log of each run should be kept and discussion of the strategies adopted held periodically. In this way, students can get some sense of the effect of each of the variables on the chances of winning the election. The instructor, who should be familiar with the assumptions built into the program as well as with contemporary politics, can explain some of the reasons for the outcome.

Election could also be tailored to incorporate the characteristics of a specific election or district. The students could research the amount of money spent by the candidates as well as the size of the district and its political composition. This research could be used to alter the program. This exercise is more demanding than the first, but it can be rewarding as it teaches students how to analyze American politics, how to do historical and political research,

DOS	Lockout Break	Restore Break
TRSDOS 2.3	POKE 23886,0	POKE 23886,1
NEWDOS 2.1	POKE 23461,0	POKE 23461,1
NEWDOS 80	POKE 19408,0	POKE 19408,1

Table 1

Program Listing

```
10 '
20 CLS:PRINTCHR$(23):PRINT,,"ELECTION"
30 PRINT:PRINT:PRINTTAB(8)"BY ROBERT JACOBS"
40 PRINT:PRINT"DRAFT - VERSION 2.1"
50 PRINT:PRINTTAB(8)"OCTOBER, 1981"
60 DIMHL$(30):DIMPH$(30):DIMHL(30):RANDOM:FORZ=1TO30:READHL$(Z),
HL(Z):NEXT
70 P$="$$###,###":Q$="##.#%":F=30000:VP=31:CO=290000+RND(30000)
80 GOSUB2130
90 '
100 CLS:PRINT"YOUR CONSTITUENCY HAS";CO;" VOTERS IN IT. YOU WISH
 TO WIN THE DISTRICT'S SEAT IN THE HOUSE OF REPRESENTATIVES."
110 PRINT"YOU START YOUR CAMPAIGN WITH";F;"DOLLARS. YOU MAY CHOO
SE TO"
```

Listing continues

Variable	Purpose	Appears in Line Number
CO	Size of Constituency	70 100 2450
DP	Democratic Percentage	390 400 420 430 440 450 500 510 580 590 670 2450
ET	Total Expenditure	2350 2360 2370 2390 2420
F	Current Campaign Fund Total	70 110 220 320 720 2190 2370 2380 2390 2410 2420
F2	Week's Receipts	300 310 320 710
FR	Total Fund Raising Expenditures	2220 2420 2630
MF	Media Factor	460 470 480 490 500 510
PS	Sum of Political Position Factors	290 340 350 360 370 440 450
RP	Republican Percentage	390 400 420 430 440 450 500 510 580 590 680 2450
SO	Total Souvenir Expenditure	2210 2420 2620
TE	Number of Position Changes	520 580 590 900
TV	Total Television Expenditures	2200 2420 2610
UP	Undecided Percentage	380 390 440 450 500 510 690
VP	Probable Turnout Percentage	70 340 350 360 370 380 700 2450 2480

Table 2. Election Variables

Listing continued

```
12Ø PRINT"USE ALL OR A PORTION OF IT ON THE LISTED POLITICAL ACT
IVITIES."
13Ø PRINT"YOU SHOULD BE AWARE THAT FUND-RAISING ITSELF HAS SUBST
ANTIAL"
14Ø PRINT"COSTS WHICH YOU WILL HAVE TO BEAR. YOU SHOULD SPEND YO
UR MONEY"
15Ø PRINT"SO AS TO MAXIMIZE YOUR CONTRIBUTIONS AS WELL AS YOUR S
TANDING"
16Ø PRINT"IN THE POLLS. yOU MAY FIND THAT THIS IS DIFFICULT TO D
O. AS THE"
17Ø PRINT"CAMPAIGN CONTINUES YOU WILL HAVE THE OPPORTUNITY TO CH
ANGE YOUR"
18Ø PRINT"STRATEGY SEVERAL TIMES. STRATEGY CHANGES MAY NOT BE IM
MEDIATELY"
19Ø PRINT"REFLECTED EITHER IN THE POLLS OR IN THE RATE OF CONTRI
BUTIONS."
2ØØ PRINT:PRINT"GOOD LUCK!":GOSUB213Ø
21Ø GOSUB1ØØØ
22Ø CLS:PRINT"YOU NOW HAVE";F;"DOLLARS. PLEASE INDICATE HOW MUCH
 YOU"
23Ø PRINT"WOULD LIKE TO SPEND FOR EACH OF THE FOLLOWING."
24Ø PRINT:PRINTTAB(7)"FUND-RAISING","TELEVISION ADVERTISING"
25Ø PRINTTAB(7)"NEWSPAPER ADVERTISING","RADIO ADVERTISING"
26Ø PRINTTAB(2Ø)"CAMPAIGN SOUVENIRS":GOSUB224Ø
27Ø GOSUB138Ø
28Ø '
29Ø WE=WE+1:PS=P1+P2+P3+P4+P5
3ØØ IFE1<2ØØØTHENF2=E1+E1*RND(1)ELSEIFE1<5ØØØANDE1>2ØØØTHENF2=E1
*(3+RND(2))ELSEF2=E1*(5+RND(2))
31Ø IFF2>49999THENF2=5ØØØØ-RND(5ØØ)
32Ø F=F+F2:F8=F8+F2
33Ø ONWEGOTO34Ø,34Ø,34Ø,35Ø,35Ø,35Ø,35Ø,36Ø,36Ø,36Ø
34Ø IFPS<1ØØRPS>2ØTHENVP=VP*1.Ø3ELSEVP=VP*1.Ø28:GOTO37Ø
35Ø IFPS<1ØØRPS>2ØTHENVP=VP*1.Ø4ELSEVP=VP*1.Ø38:GOTO37Ø
36Ø IFPS<1ØØRPS>2ØTHENVP=VP*1.Ø45ELSEVP=VP*1.Ø43
37Ø IFPS<6ORPS>23THENVP=VP*1.ØØ3
38Ø UP=(1ØØ-VP)/2.7+RND(2)
39Ø IFWE=1THENDP=(1ØØ-UP)*.52:RP=(1ØØ-UP)*.48
4ØØ DP=DP+HL(N1)+HL(N2):RP=RP-HL(N1)-HL(N2)
41Ø NB=HL(N1)+HL(N2)
42Ø IF(C$="R"ANDPØ$="I")OR(PØ$="C"ANDC$="D")THENRP=RP+.75:DP=DP-
.75
43Ø IF(C$="D"ANDPØ$="I")OR(PØ$="C"ANDC$="R")THENDP=DP+.75:RP=RP-
.75
44Ø IFC$="D"THENDP=DP-ABS(15-PS)/1Ø:RP=1ØØ-UP-DP
45Ø IFC$="R"THENRP=RP-ABS(15-PS)/1Ø:DP=1ØØ-UP-RP
46Ø MF=Ø:IFE2>18ØØØMF=MF+E2/18ØØØELSEMF=MF-1.5
47Ø IFE3>12ØØØTHENMF=MF+E3/12ØØØELSEMF=MF-1
48Ø IFE4>3ØØØTHENMF=MF+.5ELSEMF=MF-.7
49Ø IFE5>=75ØTHENMF=MF+.1ELSEMF=MF-.2
5ØØ IFC$="D"THENDP=DP+MF:RP=1ØØ-DP-UP
51Ø IFC$="R"THENRP=RP+MF:RP=1ØØ-DP-UP
52Ø IFTE>3CLSELSE61Ø
53Ø PRINTCHR$(23)"IT LOOKS AS THOUGH YOU'VE"
54Ø PRINT"CHANGED YOUR MIND ONCE TOO"
55Ø PRINT"OFTEN. THE PRESS CHARGES YOU"
56Ø PRINT"WITH INCONSISTENCY - OR WORSE!"
57Ø PRINT"YOU LOSE ONE PERCENT!"
58Ø IFC$="R"THENRP=RP-1:DP=DP+1:TE=Ø
59Ø IFC$="D"THENRP=RP+1:DP=DP-1:TE=Ø
6ØØ GOSUB213Ø
61Ø IFWE=1ØGOTO243Ø
62Ø '
63Ø CLS:PRINTCHR$(23)
64Ø PRINT@1Ø,"STAFF REPORT, WEEK";WE
65Ø FORZ=11TO1Ø6:SET(Z,3):NEXT
66Ø PRINT@196,"POLL RESULTS
67Ø PRINTTAB(7)"DEMOCRATIC:";:PRINTUSINGQ$;DP
68Ø PRINTTAB(7)"REPUBLICAN:";:PRINTUSINGQ$;RP
69Ø PRINTTAB(7)"UNDECIDED: ";:PRINTUSINGQ$;UP
7ØØ PRINTTAB(7)"PROBABLE TURNOUT:";:PRINTUSINGQ$;VP
71Ø PRINT@516,"WEEK'S RECEIPTS:";:PRINTUSINGP$;F2
72Ø PRINT@58Ø,"CAMPAIGN FUND:  ";:PRINTUSINGP$;F
73Ø PRINT@644,"SPENT TO DATE:  ";:PRINTUSINGP$;F1
74Ø PRINT@7Ø8,"TIME UNTIL ELECTION:";1Ø-WE;"WEEKS"
75Ø FORZ=11TO1Ø6:SET(Z,38):NEXT:GOSUB213Ø
76Ø '
77Ø CLS:PRINT"HERE IS A DIGEST OF THE WEEK'S MOST IMPORTANT NEWS
 AS"
78Ø PRINT"IT RELATES TO THE CAMPAIGN:"
79Ø N1=RND(3Ø):IFPH$(N1)<>""GOTO79Ø
8ØØ N2=RND(3Ø):IFN2=N1GOTO8ØØ
81Ø IFPH$(N2)<>""THENGOTO8ØØ
82Ø PH$(N1)=HL$(N1):PH$(N2)=HL$(N2)
```

Listing continues

and how to rewrite a Basic computer program.

Since each run of Election lasts only 15 or 20 minutes, a systematic approach to it would be practical. Students will profit from such an approach since they can vary strategy decisions to be tested from one run to the next.

If anyone develops any other educational uses for this program, please let me know about them.

Altering Election 2.1

Table 2 contains a list of variables you will most likely want to alter.

To change the size of the constituency, alter CO in line 70. As CO is set, it is the approximate size of a Congressional District. If you set up the program for a particular district, you can use the population of that district. For non-Congressional elections, you would also have to change the text appearing in lines 100–190.

The initial campaign fund, F, is set to $30,000 in line 70. E1 through E5, ET, F1 and F4 are temporary variables that carry items such as subtotals and the amount of money spent on each of the media. Assumptions regarding the effect of these expenditures appear in lines 460–490. For example, E3 is the amount spent on television each week. In line 470 you see that if a minimum of $12,000 has not been spent on television advertising that week, the media factor, MF, is reduced by one. If television spending is greater than $12,000, MF is incremented by one plus the amount spent divided by 12,000. Variable MF carries each of the media expenditure factors into the candidates vote total in lines 500 and 510.

The initial Democratic "edge" is established by line 390.

The news headlines appear as data statements in lines 2780 to the end. The number after each headline is the percentage to be added to the Democratic side and subtracted from the Republican side. This calculation occurs in line 400. The text of the headlines and their political effect can be adjusted to work in either direction. The program goes through two-thirds of the headlines in each run, so the general direction, or balance, of the news, can be controlled by the programmer. More headlines can be added in machines that have more than 16K memory by adding data statements and adjusting the high number of the loop in line 60 to reflect the number of headlines to be read into HL$(Z). For example, if there were 40

Listing continued

```
830 PRINT:PRINT:PRINT:PRINTPH$(N1)
840 PRINT:PRINT:PRINT:PRINTPH$(N2)
850 GOSUB2130
860 '
870 CLS:PRINT:PRINT:PRINTCHR$(23):PRINT"DO YOU WISH TO CHANGE AN
Y OF":PRINT"YOUR ANNOUNCED POSITIONS?  (Y/N)";:GOSUB2140
880 IFZ$="Y"THENGOTO900
890 GOTO2160
900 TE=TE+1:TC=TC+1:CLS:PRINT"ON WHICH ISSUE DO YOU WISH TO CHAN
GE YOUR POSITION?"
910 PRINT:PRINT:PRINTTAB(5)"(1) SOCIAL WELFARE","CURRENT POSITIO
N IS # ";P1
920 PRINT:PRINTTAB(5)"(2) THE ECONOMY","CURRENT POSITION IS # ";
P2
930 PRINT:PRINTTAB(5)"(3) BUSING",,"CURRENT POSITION IS # ";P3
940 PRINT:PRINTTAB(5)"(4) WOMEN'S RIGHTS","CURRENT POSITION IS #
 ";P4
950 PRINT:PRINTTAB(5)"(5) FOREIGN POLICY","CURRENT POSITION IS #
 ";P5
960 PRINT:GOSUB2150:Z=VAL(Z$):IFZ<0ORZ>5GOTO900ELSEONZGOSUB1510,
1630,1750,1870,1990
970 CLS:PRINT@192,"WOULD YOU LIKE TO MAKE ADDITIONAL POSITION CH
ANGES? (Y/N)";:GOSUB2140
980 IFLEFT$(Z$,1)="Y"THENGOTO900ELSEGOTO2160
990 '
1000 CLS:PRINT"YOU WILL NOW HAVE THE OPPORTUNITY TO MAKE SOME IN
ITIAL POLITICALCHOICES. YOU MAY BE AN INCUMBENT OR A CHALLENGER,
 A DEMOCRAT OR REPUBLICAN. IF YOU WISH, THE COMPUTER WILL CHOOSE
 FOR YOU."
1010 PRINT"AFTER MAKING YOUR INITIAL CAMPAIGN FUND DISBURSEMENTS
 YOU"
1020 PRINT"WILL BE ASKED TO MAKE SOME POLICY CHOICES WHICH WILL
LOCATE"
1030 PRINT"YOU ON AN IDEOLOGICAL SCALE THUS HELPING DETERMINE WH
AT"
1040 PRINT"SHARE YOU WILL CAPTURE OF THE VOTE OF THE MAJOR GROUP
S IN "
1050 PRINT"YOUR CONSTITUENCY."
1060 PRINT:PRINT
1070 PRINTTAB(5)"WOULD YOU LIKE TO BE A (D)EMOCRAT OR A (R)EPUBL
ICAN -"
1080 PRINTTAB(8)"OR WOULD YOU LIKE THE (C)OMPUTER TO CHOOSE?":PR
INTTAB(8)"PRESS <D>,
<R>, OR <C>.":GOSUB2140
1090 C$=Z$:IFC$="C"THENIFRND(2)=1THENC$="D"ELSEC$="R"
1100 IFC$<>"R"ANDC$<>"D"THEN1000
1110 IFC$="R"GOTO1190
1120 CLS:PRINT:PRINT:PRINT:PRINTTAB(10)"YOU'VE MADE AN EXCELLENT
 CHOICE.
1130 PRINTTAB(10)"NOW YOU CAN WRAP YOURSELF IN THE MANTLE
1140 PRINTTAB(10)"OF SUCH OTHER DEMOCRATS AS JEFFERSON,"
1150 PRINTTAB(10)"WILSON, ROOSEVELT, AND KENNEDY BY SELECTING"
1160 PRINTTAB(10)"POSITIONS YOU THINK APPROPRIATE TO THE"
1170 PRINTTAB(10)"DEMOCRATIC PARTY."
1180 GOSUB2130:GOTO1250
1190 CLS:PRINT:PRINT:PRINT:PRINTTAB(10)"A FINE CHOICE! NOW YOU M
AY"
1200 PRINTTAB(10)"ATTEMPT TO EMULATE SUCH REPUBLICANS AS"
1210 PRINTTAB(10)"LINCOLN, ROOSEVELT, AND EISENHOWER BY
1220 PRINTTAB(10)"SELECTING POLITICAL POSITIONS WHICH ARE
1230 PRINTTAB(10)"APPROPRIATE TO THE PARTY."
1240 GOSUB2130
1250 CLS:PRINT"WOULD YOU PREFER TO BE AN INCUMBENT OR CHALLENGER
?"
1260 PRINT"BEFORE DECIDING YOU SHOULD BE AWARE THAT IN AMERICAN
ELECTIONS"
1270 PRINT"INCUMBENTS ARE NORMALLY RE-ELECTED MORE THAN 60% OF T
HE TIME."
1280 PRINT"IN 'BAD' TIMES THIS PERCENTAGE MAY DROP SHARPLY. THE
'NATURE'"
1290 PRINT"OF THE TIMES FOR THIS SIMULATION ARE DETERMINED PARTL
Y BY"
1300 PRINT"THE PROGRAMMER(PESSIMISTIC ON THURSDAYS AND OPTIMISTI
C ON"
1310 PRINT"MONDAYS) AND IN PART BY THE NEWS HEADLINES WHICH ARE
SELECTED"
1320 PRINT"BY A RANDOM NUMBER GENERATOR."
1330 PRINT@715,"(I)NCUMBENT OR (C)HALLENGER?";:GOSUB2140
1340 P0$=Z$:IFP0$<>"I"ANDP0$<>"C"THENGOTO1250
1350 IFP0$="I"CLS:PRINTCHR$(23):PRINT@260,"PLAYING IT SAFE MAY O
R MAY NOT":PRINT@346,"HELP":PRINT@530,"GOOD LUCK!":GOTO1370
1360 CLS:PRINTCHR$(23):PRINT@260,"AN UPHILL STRUGGLE. GOOD LUCK!
"
1370 GOSUB2130:RETURN
1380 '
1390 CLS:PRINT"YOU WILL NOW SELECT POSITIONS ON FIVE POLITICAL I
```

Listing continues

✓See List of Advertisers on Page 563

headlines, the relevant portion of line 30 would say FOR Z = 1 TO 40:.

PS represents the total effect of the positions selected. The program sums the numbers of the student's policy choices. The low-end choices, 1 and 2, tend to be conservative, and the high-end choices, 4 and 5, tend to be liberal. The appropriate party is incremented or decremented in lines 440 and 450.

The size of the turnout, VP, is directly tied to the degree of controversy injected into the campaign by these choices. The constants that control this are found in lines 340–370. The issues can be changed, and different weights can be assigned to the policy choices. In lines 440–450, the effect of ideology could be increased as a whole by lowering the constant 10 by a small amount, or you can use a particular issue as a key issue by adding to these lines. For example, if issue 2 were given extra weight or effect, these lines could become:

```
440 IF C$ = "D" THEN
 DP = DP-(ABS(15 – PS)/10)-ABS(3 – P2):
RP = 100-UP-DP

450 IF C$ = "R" THEN
RP = RP-(ABS(15 – PS)/10)-ABS(3-P2):
DP = 100-UP-RP
```

If this technique is selected, you should adjust the content of the news headlines to reflect the increased importance of the selected issue.

The ideological range of each issue can best be adjusted at the point in the program where the choices are made—lines 1610, 1730, 1850, 1970, and 2100. The position value is set equal to the value of Z$. At this point, you could alter the range by adding or subtracting an appropriate amount, for example:

```
1610 GOSUB 2150:P1 = VAL(Z$)-2: IFP1
<5ORP1> 1GOTO1510
```

This has the effect of moving the responses on issue 1 well toward the liberal side; adding 2 to P1 would move them toward the conservative side. The program is structured to accept a response of 3 to each position to indicate roughly the ideological center of the constituency. You should change the text of the positions, instead of the arithmetic of the program, since numerical changes, if too large, will have distorting effects that are multiplied over the 10 weeks of the campaign. ■

Robert Jacobs (Central Washington University, Ellensburg, WA 98926) is a professor of political science.

Listing continued

```
SSUES. THESE"
1400 PRINT"ARE SOCIAL WELFARE, THE ECONOMY, BUSING, WOMEN'S RIGH
TS AND FOREIGN POLICY.  IN SELECTING YOUR PUBLIC POSITIONS YOU S
HOULD BE"
1410 PRINT"AWARE THAT YOU WILL BE EVALUATED BY THE VOTERS IN TER
MS OF YOUR"
1420 PRINT"CONSISTENCY AS WELL AS THE EXTENT TO WHICH YOUR POLIC
Y CHOICES"
1430 PRINT"ARE IN AGREEMENT WITH THEIRS. AS THE CAMPAIGN PROGRES
SES YOU"
1440 PRINT"MAY BE ASKED TO DISCUSS THESE ISSUES FURTHER, PERHAPS
 IN THE"
1450 PRINT"LIGHT OF ANY NEWS WHICH MAY DEVELOP."
1460 PRINT:PRINT:PRINT"FOUR OR FIVE POLICY POSITIONS ARE SUGGEST
ED TO YOU BY"
1470 PRINT"YOUR STAFF ON EACH OF THE ISSUES. CHOOSE YOUR STAND B
Y"
1480 PRINT"ENTERING THE NUMBER OF THE POSITION YOU PREFER."
1490 GOSUB2130:GOSUB1510:GOSUB1630:GOSUB1750:GOSUB1870:GOTO1990
1500 RETURN
1510 '
1520 CLS:PRINT"HERE ARE THE PROPOSED POSITIONS ON SOCIAL WELFARE
:"
1530 :PRINT:PRINT"(1)  I THINK THAT THE ONLY PRACTICAL WAY TO DE
AL WITH THE"
1540 PRINTTAB(5)"WELFARE SYSTEM IS TO IMPOSE SEVERE BUDGET CUTS.
"
1550 PRINT:PRINT"(2)  WE SHOULD DO EVERYTHING WE CAN TO END WELF
ARE FRAUD."
1560 PRINT:PRINT"(3)  WE MUST CONTINUE TO CARE FOR DEPENDENT CHI
LDREN, THE SICK"
1570 PRINTTAB(5)"AND THE ELDERLY IN A CONTEXT OF FISCAL RESPONSI
BILITY"
1580 PRINT:PRINT"(4)  WE MUST INCREASE THE SERVICES WE OFFER THE
 POOR."
1590 PRINT:PRINT"(5)  THE POOR ARE THE VICTIMS OF AN UNFEELING S
YSTEM; THEY MUST"
1600 PRINTTAB(5)"BE AIDED TO THE MAXIMUM POSSIBLE EXTENT."
1610 GOSUB2150:P1=VAL(Z$):IFP1>5ORP1<1GOTO1510
1620 RETURN
1630 '
1640 CLS:PRINT"HERE ARE THE PROPOSED POSITIONS ON THE ECONOMY:"
1650 PRINT:PRINT"(1)  IMMEDIATE REDUCTION OF TAXES, BOTH CORPORA
TE AND PERSONAL;
1660 PRINTTAB(5)"ADDITIONAL TAX INCENTIVES TO BUSINESS."
1670 PRINT:PRINT"(2)  BALANCED BUDGET EVEN IF SOME PROGRAMS HAVE
 TO BE CUT."
1680 PRINT:PRINT"(3)  SAME LEVEL OF EXPENDITURE AS LAST YEAR; NO
 ADDED TAXES."
1690 PRINT:PRINT"(4)  SOME NEW PROGRAMS TO HELP COPE WITH UNEMPL
OYMENT;"
1700 PRINTTAB(5)"NO NEW TAXES."
1710 PRINT:PRINT"(5)  MAJOR PROGRAMS TO END UNEMPLOYMENT FINANCE
D BY NEW TAXES;"
1720 PRINTTAB(5)"LOWER CEILING ON INTEREST RATES."
1730 GOSUB2150:P2=VAL(Z$):IFP2>5ORP2<1GOTO1630
1740 RETURN
1750 '
1760 CLS:PRINT"HERE ARE SOME PROPOSALS ON THE BUSING ISSUE:"
1770 PRINT:PRINT"(1)  FAVOR AN ANTI-BUSING CONSTITUTIONAL AMENDM
ENT"
1780 PRINT:PRINT"(2)  NO FEDERAL FUNDS TO BE USED FOR BUSING. A
RIDER
1790 PRINTTAB(5)"SHOULD BE ATTACHED TO THE APPROPRIATIONS BILL."

1800 PRINT:PRINT"(3)  BUSING IS A LOCAL ISSUE; THE STATES CAN DE
AL WITH"
1810 PRINTTAB(5)"THE COURTS AND WORK IT OUT."
1820 PRINT:PRINT"(4)  BUSING IS CONSTITUTIONALLY MANDATED AND IS
 PART"
1830 PRINTTAB(5)"OF THE LAW OF THE LAND. THE FEDERAL GOVERNMENT
MUST"
1840 PRINTTAB(5)"CONTINUE TO SUPPORT IT."
1850 GOSUB2150:P3=VAL(Z$):IFP3>4ORP3<1GOTO1750
1860 RETURN
1870 '
1880 CLS:PRINT"HERE ARE POSITIONS ON WOMEN'S RIGHTS:"
1890 PRINT:PRINT"(1)  SUPPORT ANTI-ABORTION AMENDMENT, DEFEAT ER
A."
1900 PRINT:PRINT"(2)  CONTINUE THE BAN ON FEDERAL FUNDING OF ABO
RTIONS;"
1910 PRINTTAB(5)"DEFEAT THE ERA"
1920 PRINT:PRINT"(3)  ATTEMPT TO PRESERVE TRADITIONAL FAMILY STR
UCTURE."
1930 PRINT:PRINT"(4)  ERA IS EFFECTIVELY LOST ANYWAY, BUT LEGAL
```

Listing continues

Listing continued

```
BARRIERS TO"
1940 PRINTTAB(5)"EQUAL RIGHTS FOR WOMEN SHOULD BE ABOLISHED."
1950 PRINT:PRINT"(5)  SUPPORT ERA AND RESTORE FEDERAL MEDICAL AS
SISTANCE FOR"
1960 PRINTTAB(5)"ABORTIONS FOR THOSE WOMEN WHO CAN'T OTHERWISE A
FFORD THEM."
1970 GOSUB2150:P4=VAL(Z$):IFP4>5ORP4<1GOTO1870
1980 RETURN
1990 '
2000 CLS:PRINT"THESE ARE THE FOREIGN POLICY POSITIONS:"
2010 PRINT:PRINT"(1)  ECONOMIC SANCTIONS AGAINST THE U.S.S.R; AB
ANDON SALT"
2020 PRINTTAB(5)"AND GIVE MORE SUPPORT TO ANTI-COMMUNIST REGIMES
."
2030 PRINT:PRINT"(2)  CUT GRAIN AND HIGH-TECHNOLOGY EXPORTS TO T
HE SOVIET UNION;"
2040 PRINTTAB(5)"ADOPT A TOUGHER STANCE TOWARDS SALT."
2050 PRINT:PRINT"(3)  TRADE WITH THE SOVIET UNION UNION TO CONTI
NUE AT PRESENT"
2060 PRINTTAB(5)"LEVELS; MAINTAIN EXISTING SALT AGREEMENTS AND S
UPPORT"
2070 PRINTTAB(5)"FURTHER DISARMAMENT AGREEMENTS."
2080 PRINT:PRINT"(4)  ABANDON SUPPORT FOR AUTHORITARIAN REGIMES;
 CUT MILITARY
2090 PRINTTAB(5)"SPENDING AND ADOPT STRONG HUMAN RIGHTS POLICY."

2100 GOSUB2150:P5=VAL(Z$):IFP5>4ORP5<1GOTO1990
2110 RETURN
2120 END
2130 PRINT@900,"PRESS <ENTER> TO CONTINUE";
2140 Z$="":Z$=INKEY$:IFZ$=""THEN2140ELSERETURN
2150 PRINT"WHICH DO YOU CHOOSE?";:GOSUB2140:RETURN
2160 '
2170 CLS:PRINT,"WEEK";WE+1;"OF THE CAMPAIGN"
2180 PRINT:PRINT"   NOW YOU SHOULD DECIDE HOW MUCH OF YOUR WAR C
HEST TO"
2190 PRINT"SPEND ON EACH CAMPAIGN ACTIVITY FOR THIS WEEK. yOUR B
ALANCE IS";:PRINTUSINGP$;F;:PRINT"."
2200 PRINT"THUS FAR YOU HAVE SPENT";:PRINTUSINGP$;TV;:PRINT" ON
TELEVISION,";:PRINTUSINGP$;RA
2210 PRINT"ON RADIO,";:PRINTUSINGP$;NE;:PRINT" ON NEWSPAPER ADVE
RTISING,";:PRINTUSINGP$;SO;:PRINT" FOR"
2220 PRINT"SOUVENIRS, AND";:PRINTUSINGP$;FR;:PRINT" ON FUND RAIS
ING.":GOSUB2240
2230 GOTO280
2240 PRINT:PRINT:PRINTTAB(7)"ENTER EXPENDITURE FOR FUND-RAISING"
;:INPUTE1
2250 PRINTTAB(7)"ENTER EXPENDITURE FOR TELEVISION";:INPUTE2
2260 PRINTTAB(7)"ENTER EXPENDITURE FOR RADIO";:INPUTE3
2270 PRINTTAB(7)"ENTER EXPENDITURE FOR NEWSPAPERS";:INPUTE4
2280 PRINTTAB(7)"ENTER EXPENDITURE FOR SOUVENIRS";:INPUTE5
2290 CLS::PRINT"YOU HAVE SPENT YOUR CAMPAIGN MONEY AS FOLLOWS:"
2300 PRINT:PRINT:PRINTTAB(15)"FUND-RAISING:";:PRINTUSINGP$;E1
2310 PRINTTAB(15)"TELEVISION:   ";:PRINTUSINGP$;E2
2320 PRINTTAB(15)"RADIO:        ";:PRINTUSINGP$;E3
2330 PRINTTAB(15)"NEWSPAPERS:   ";:PRINTUSINGP$;E4
2340 PRINTTAB(15)"SOUVENIRS:    ";:PRINTUSINGP$;E5
2350 FORZ=30TO79:SET(Z,25):NEXT:ET=E1+E2+E3+E4+E5
2360 PRINT:PRINTTAB(15)"     TOTAL:    ";:PRINTUSINGP$;ET
2370 PRINT:PRINT:IFET>FTHENPRINT"SORRY; YOU HAVE ONLY";F;" DOLLA
RS IN THE TREASURY"ELSEGOTO2390
2380 INPUT"HIT <ENTER> TO REDO YOUR BUDGET";Z:CLS:PRINT"TOTAL AV
AILABLE IS";:PRINTUSINGP$;F:GOTO2240
2390 F4=F-ET:PRINT"THESE EXPENDITURES WILL LEAVE";F4;"DOLLARS"
2400 PRINT"IN YOUR CAMPAIGN TREASURY. IS THIS WHAT YOU WISH(Y/N)
";:GOSUB2140
2410 IFZ$<>"Y"THENCLS:PRINT"TOTAL FUNDS AVAILABLE";:PRINTUSINGP$
;F:GOTO2240
2420 F=F4:F1=F1+ET:FR=FR+E1:TV=TV+E2:RA=RA+E3:NE=NE+E4:SO=SO+E5:
RETURN
2430 '
2440 CLS:PRINT"HERE ARE THE RESULTS OF THE ELECTION:"
2450 XC=100*DP/(DP+RP):QC=100*RP/(RP+DP):VO=CO*VP/100
2460 PRINT:PRINT:PRINTTAB(5)"REPUBLICAN:",INT(VO*QC/100),:PRINTU
SINGQ$;QC
2470 PRINT:PRINTTAB(5)"DEMOCRATIC:",INT(VO*XC/100),:PRINTUSINGQ$
;XC
2480 PRINT:PRINTTAB(5)"TOTAL VOTE:",VO,:PRINTUSINGQ$;VP
2490 IFXC>QCANDC$="D"ORC$="d"GOTO2510
2500 IFQC>XCANDC$="R"THENGOTO2510ELSE2540
2510 PRINT:PRINT"YOU'VE WON! CONGRATULATIONS ON YOUR ELECTION. T
HE"
2520 PRINT"CHANCES ARE THAT YOUR PROBLEMS ARE ONLY BEGINNING. PR
ESS"
2530 PRINT"<ENTER> TO SEE A SUMMARY OF YOUR CAMPAIGN.":GOSUB2140
```

Listing continues

TAS Adventure Series Package I

By: Gary Heimburger and Gary Robinson

(This adventure package is the first to be marketed that was created by The Adventure System! Look for more fantastic adventures coming soon! Each TAS adventure package includes TWO complete adventures for one low price!)

Adventure 1: Isles of Doom - Part I

The Scenario: A small South-sea island chain. Unfortunate navigation left you stranded on one island.

The Mission: (if you can get through the shark-infested waters to the island!) The island was a scene of activity during WWII. Scattered over the island are components that will allow you to fix an old radio transmitter. Can you fix it and tap out your SOS before falling prey to the elements of doom?

Adventure 2: Lost City of Gold

The Scenario: You're on a mountain climbing expedition in South America (in Peru, near Lake Titicaca). After straying from the beaten path, you discover a cave that eventually leads to a forgotten part of the lost Incan city. Grab your pitons.

The Mission: There are plenty of treasures if you can safely collect them and retrace your steps to the simple mountain hut. Be forwarned that most Incan cities contain many perils, especially for the uninitiated...seems Incan priests were rather paranoid. Requires 32K.

TAS Adventure Packages are $15.95 on tape and $19.95 on disk. All media is unprotected! TAS Adventures allow you to SAVE up to 10 versions of the game in progress! Please specify Model I or III.

***

Get in on the action! The Adventure System editor is now completely in Z80 and allows easy interactive debugging. Requires 48K. Specify Tape or Disk, please. Only $39.95 for the complete system AND three adventures!

STOPPER!

The BASIC Breakpointer

By Roxton Baker

Author of "TRAKCESS"

Stopper is a unique machine language utility that allows interactive program debugging when using Level II or Disk BASIC on the Model I and III. Stopper was designed by Roxton to help debug a new version of TRAKCESS that will be announced shortly, because there wasn't anything on the market that would provide the features needed.

Some of Stopper's commands: Set a breakpoint with a "hit" count, either when a line has been executed "n" times or a variable is equal a certain value; or not equal a certain value; breakpoints and variables may be viewed and modified at any time; selective execution of program statements and lines; display only the portion of the line where the error occurred; TRON functions are improved and may be routed to printer; set a "tolerance" on single and double precision numbers (to accommodate the inaccurate ROM routines and more. Naturally single stepping is fully supported.

With STOPPER you will know exactly where the error is and why it occurred. Program execution is under YOUR control!

A special offer from the author and TAS: Stopper is $20 on tape or disk, BUT you are licensed to make up to four additional copies, making the cost for this valuable utility only $4. Support co-op purchases by ordering today. BASIC debugging will never be easier, nor a better utility any cheaper. Extra manuals and Z80 source code are available.

LDOS Users!

A special utility package just for YOU:

DRIVE-5

Four drives not enough? DRIVE-5 is a memory based disk emulator which allows you to configure a fifth drive IN MEMORY! No moving parts, so access is FAST! In a two drive system, FORMAT and BACKUP can be placed on DRIVE-5 (with /SYS modules) to allow multiple backups with less disk swapping.

LOWDUMP

LOWDUMP allows you to DUMP files from anywhere in memory, including ROM and video RAM! LOWDUMP allows all parameters as the LDOS DUMP command, including ASCII, for dumping files without loader information present.

SYSGEN

SYSGEN/CMD allows you to save one or more SYSGEN files as normal /CMD files. Keep several configurations on the same disk and reload as necessary by simply typing the filename. Especially useful when the preferred configuration consumes so much memory that large programs will not execute properly. SYSGEN/CMD may also be executed from a running BASIC program. When the resultant /CMD file is executed, the BASIC program will continue with the next statement, all program variables intact!

UNREPAIR

Model I TRSDOS cannot read LDOS disks due to the F8 DAM used for directory sectors. UNREPAIR will rewrite the directory track of a single density disk using the FA DAM that will be recognized by Model I TRSDOS. UNREPAIR should run with any Model I DOS and the Radio Shack E/I.

An extra BONUS! The TAS LDOS Utility pack includes Z80 source code (on disk) of most of the above utilities (the most informative). The LDOS Utility Pak is $49.95. This package was written by Les Mikesell.

THE ALTERNATE SOURCE 69

EFFECTIVE COMMUNICATION

is easy with Modem80. There is absolutely no need to pay any more for a top-quality communication package. Modem80 can help your TRS-80 Model I and III talk to a wide variety of machines. The Modem80 translation table is easily modified to accommodate unusual protocol. Modem80 allows file uploading and downloading. XON and XOFF are supported to allow transmission of files larger than memory. A special protocol (compatible with the CP/M public domain program MODEM) allows error-free transmission without converting to ASCII. XMODEM, a special version of MODEM is included so that you may give your friends a copy to take advantage of faster error-free transmission. A host program is included at no extra charge to allow accessing your unattended TRS-80 from a remote location. All "local" parameters (baud rate, stop bits, parity, duplex and more) can be changed quickly and easily with a minimum of keystrokes. For specialized applications, one file may be transmitted while a different file is simultaneously being received. DOS commands may be executed while within Modem80. Special patches insure compatibility with all popular Model I and III operating systems. The previous screen is restored after DOS commands. Modem80 also allows disabling screen scroll to permit transmission speeds at up to 9600 baud (under direct connect situations). A special patch sheet is available for people wanting to use the Micronet Dow Jones service. Also available, on request, is a LYNX modem version that allows operation on a Model III even if RS-232 is installed. All this and many more features are what's making Modem80 one of our most popular programs! Oh, yes, the price helps too: $39.95 complete including documentation in a three-ring binder. Modem80 is authored by Les Mikesell.

BRAND NEW!

By Bruce Hansen

An exciting new arcade game! Threshold-Nine is easy, but only the experienced reach Threshold-Zero. As you progress to the faster levels, the playing field decreases! Excellent graphics and sound, joystick compatible, all Z80 programming! Supplied in a "back-uppable" format (disk users will need to copy to a TRSDOS system diskette). High scores are saved on the disk version. Includes a special "pause" feature (for nature calls and other interrupt processing). Works on Models I and III cassette ($15.95) and disk ($19.95).

FREE to the first 100 customers who order Threshold-Zero: A mini-poster of the Threshold-Zero Cover Art! Order today and find out why Threshold-Zero is already one of our best selling games!

Order Form

Please send me the following:

- ☐ TAS Adventure Package I (tape) $15.95
- ☐ TAS Adventure Package I (disk) $19.95
- ☐ The Adventure System (tape) $39.95
- ☐ The Adventure System (disk) $39.95
- ☐ LDOS Enhancement Package $49.95
- ☐ Threshold-Zero (tape) $15.95
- ☐ Threshold-Zero (disk) $19.95
- ☐ Stopper $20.00
- ☐ Modem80 $39.95
- ☐ **TAS Subscription $24.00**

Please add $3 to your total order for shipping (except TAS subscribers). We ship UPS unless you specify otherwise.

Name: ______________________
Address: ______________________
Address: ______________________
City: ______________________
State, Zip: ______________________

Mail order to:

The Alternate Source
704 N. Pennsylvania Ave.
Lansing, Michigan 48906
Ph. (800) 248-0284
Ph. (517) 482-8270

Master Card, Visa and COD ($1.75 extra) are welcome!

Listing continued

```
:GOTO2580
2540 PRINT:PRINT"YOU'VE LOST; PLEASE ACCEPT OUR REGRETS. PERHAPS
"
2550 PRINT"IT WILL CONSOLE YOU TO KNOW THAT YOUR OPPONENT'S TROU
BLES"
2560 PRINT"ARE JUST BEGINNING.  PRESS <ENTER> TO SEE A SUMMARY"
2570 PRINT"OF THE CAMPAIGN.":GOSUB2140
2980 DATA"TWELVE ARRESTED IN WELFARE FRAUD RING",-2
2990 DATA"STUDY SHOWS SCHOOLCHILDREN IN BUSES MORE THAN TWO HOUR
S DAILY ON AVERAGE",-3
3000 DATA"STUDY SHOWS NEW BUS ROUTES CUT AVERAGE TRIP TO SCHOOL
BY FIFTEEN PERCENT",1
3010 DATA"YOUR OPPONENT TO MAKE THREE MAJOR TELEVISION ADDRESSES
 THIS WEEK",0
3020 DATA"YOUR OPPONENT BEGINS PRINT MEDIA SATURATION CAMPAIGN",
0
3030 DATA"YOUR OPPONENT TAKES MIDDLE GROUND ON BUSING ISSUE",-1
3040 DATA"YOUR OPPONENT USING DOZENS OF RADIO SPOTS THIS WEEK",0

3050 DATA"INFLUENTIAL COLUMNIST SAYS YOU LOOK 'TIRED;' YOUR OPPO
NENT HIKES FORTY MILES",0
3060 DATA"WELL-KNOWN NEWSMAN REPORTS THAT YOUR OPPONENT IS FALLI
NG OFF IN THE POLLS",0
3070 DATA"OLD CHARGES OF TAX IRREGULARITIES SURFACE AGAINST YOU"
,0
2580 CLS:PRINTTAB(30);"SUMMARY":FORZ=60TO74:SET(Z,3):NEXT
2590 PRINT:PRINT"   DURING THE COURSE OF THIS ELECTION CAMPAIGN"

2600 PRINT"YOU INCURRED EXPENDITURES AS FOLLOWS:"
2610 PRINT:PRINT"TELEVISION:";:PRINTUSINGP$;TV;:PRINT,"RADIO:
";:PRINTUSINGP$;RA
2620 PRINT"NEWSPAPERS:";:PRINTUSINGP$;NE;:PRINT,"SOUVENIRS:";:PR
INTUSINGP$;SO
2630 PRINT:PRINT"YOU RAISED";F8;"DOLLARS AT THE COST OF";FR;"DOL
LARS. THE"
2640 PRINT"BEST RATIO YOU COULD HAVE ACHIEVED IS ABOUT 5 TO 1."
2650 IFNB>0THEND$="DEMOCRATIC"ELSED$="REPUBLICAN"
2670 IFNB=0PRINT"DURING THIS RUN OF THE SIMULATION THE NEWS WAS
NEUTRAL.":GOTO2710
2680 PRINT"THE NEWS DURING THIS RUN OF THE SIMULATION FAVORED TH
E"
2690 PRINTD$;" SIDE BY";ABS(NB);"PER CENT."
2700 Z$="":IFTC>1THENZ$="S"
2710 IFTC=0GOTO2730ELSEPRINT"YOU CHANGED YOUR POSITION";TC;"TIME
";Z$;", THUS LAYING YOURSELF"
2720 PRINT"OPEN TO THE CHARGE OF INCONSISTENCY.":GOTO2750
2730 PRINT"YOU DID NOT CHANGE YOUR POSITIONS AT ALL DURING THIS
RUN"
2740 PRINT"OF THE SIMULATION, THUS AVOIDING IMPUTATIONS OF INCON
SISTENCY."
2750 PRINT@900,"ANOTHER TRY? (Y/N)";:GOSUB2140
2760 IFZ$="Y"RUNELSEEND
2770 '
2780 DATA"SOVIETS INCREASE MILITARY FORCES IN AFGHANISTAN",-2
2790 DATA"SOVIETS INVADE POLAND, CRUSH FREE LABOR MOVEMENT",-4
2800 DATA"SOVIET UNION WITHDRAWS FROM AFGHANISTAN",2
2810 DATA"U.S. SUPPORTED REGIME IN EL SALVADOR IMPRISONS TWO AME
RICAN       BISHOPS",2
2820 DATA"DISSIDENTS TORTURED BY U.S. SUPPORTED GOVERNMENT IN GU
ATEMALA",1
2830 DATA"OIL COMPANIES RAISE GASOLINE PRICES BY 4%",2
2840 DATA"DEPARTMENT OF ENERGY ANNOUNCES NEW FINDS OF NATURAL GA
S",1
2850 DATA"UNEMPLOYMENT RISES TO EIGHT PERCENT",2
2860 DATA"INFLATION RATE INCREASES TO 1.2% FOR LAST MONTH",2
2870 DATA"INFLATION RATE SAID TO DECREASE TO .4% LAST MONTH",-1
2880 DATA"JOBLESS RATE DROPS TO FIVE PERCENT",-2
2890 DATA"YOUR CAMPAIGN MANAGER ACCUSED OF HAVING A MERETRICIOUS
          RELATIONSHIP WITH HIS SECRETARY",0
2900 DATA"YOU ARE ACCUSED OF SPENDING CAMPAIGN FUNDS FOR PERSONA
L PURPOSES",0
2910 DATA"YOUR OPPONENT'S SON ARRESTED FOR DRIVING WHILE INTOXIC
ATED",0
2920 DATA"YOUR OPPONENT  FOUND TO BE RECEIVING ILLEGAL CAMPAIGN
DONATIONS",0
2930 DATA"SOME OF THE DONATIONS YOU HAVE RECEIVED ARE FOUND TO B
E ILLEGAL",0
2940 DATA"YOUR OPPONENT ANNOUNCES THAT HIS POLLS SHOW HIM LEADIN
G BY       TEN PERCENT",0
2950 DATA"PRO-CHOICE AND PRO-LIFE DEMONSTRATORS CLASH AT COURTHO
USE",0
2960 DATA"TAX RELIEF PETITION SIGNED BY 34,000; DEMAND ACTION RI
GHT AWAY",-2
2970 DATA"WELFARE MOTHERS DEMAND HIGHER BENEFITS; CLAIM CHILDREN
 STARVING",1
```

LOAD 80

To Boldly Go. . .

by Joey Robichaux

A unique application for your microcomputer, this menu-driven, Basic package will help amateur astronomers and other stargazers.

With Star Track, you can determine the position (right ascension and declination), the distance from Earth, angular size, and phase of any planet in the solar system. You can determine the positions of the sun and moon, their angular sizes and distances, and their rise and set times. You can calculate precession from the three most common epochs (1950, 1975, 2000), and determine rise and set times for any celestial object. You can also use Star Track to convert mean standard time to sidereal time, and vice versa.

Even though Star Track doesn't consider refraction or planetary perturbations when computing positions, the results are usually within a few percentage points of the actual figures. This is fine for amateur purposes, where the emphasis is on locating and observing objects.

Dictionary of Terms

Star Track introduces terms foreign to a beginner: right ascension (RA), declination (DEC), precession and epoch. The concepts involved are simple; understanding them allows you to locate any celestial object with star charts.

RA and DEC are terms concerned with locating objects in the sky (similar to latitude and longitude).

Latitude refers to how far up or down a point is from the equator, and ranges from 0-90 degrees. Zero degrees is a point on the equator, while 90 degrees north or south is either of the two poles. Latitude is expressed in de-

Program Listing

```
1 CLS:DEFDBLO-Z:DEFINTI-K,M,N:RA=.01745329:RE=23.43*RA:ID!=0
2 DIMT1(8),T2(8),T3(8),T4(8),T5(8),T6(8),T7(8),T8(8),T9(8),P$(8)
,UB(25):UA=.065709:UC=1.002743:UD=.997257
3 GOSUB7050
8 CLS
10 PRINT@20,"STAR TRACK / MASTER MENU"
25 PRINT@266,"1. Determine planet coordinates"
27 PRINT@330,"2. Determine SUN information"
28 PRINT@394,"3. Determine MOON information"
30 PRINT@458,"4. Determine precession / rise and set"
32 PRINT@522,"5. Time system conversions"
34 PRINT@586,"6. Terminate program"
50 PRINT@138,"Please select an option"
55 QQ$=INKEY$:IFQQ$<"1"ORQQ$>"6"THEN55
57 I=VAL(QQ$)
60 IFI<1THEN50
65 IFI>6THEN50
70 ONIGOTO100,190,300,400,500,170
100 CLS:PRINT@20,"STAR TRACK / PLANET TRACKER"
105 PRINT@276,"1. MERCURY":PRINT@340,"2. VENUS":PRINT@404,"3. EA
RTH"
110 PRINT@468,"4. MARS":PRINT@532,"5. JUPITER":PRINT@596,"6. SAT
URN"
115 PRINT@660,"7. URANUS":PRINT@724,"8. NEPTUNE":PRINT@788,"9. P
LUTO"
117 PRINT@852,"A. Return to main menu"
120 PRINT@138,"Please select an option"
125 QQ$=INKEY$:IFQQ$="A"THENIP=10:GOTO130:ELSEIFQQ$<"1"ORQQ$>"9"
THEN125ELSEIP=VAL(QQ$)
130 IFIP<1ORIP>10THEN120
132 IFIP=10THENCLS:GOTO10
133 IP=IP-1
134 PRINT@138,"";
135 INPUT"Enter the desired date   <MMDDYY>";D
136 GOSUB4000:IFNO=1THENGOTO134:ELSEIFIP=2THENGOTO180
137 S$=P$(IP)
140 ID=IM
145 GOSUB2200:GOSUB2300
150 IFIP<3THENGOSUB2400:ELSEGOSUB2500
160 GOSUB3000
165 GOSUB3500
166 PRINT@448,"      pressRE1> to return to last menu":PRINT"
     else press <SPACE BAR> for angular size":PRINTTAB(29)"dista
nce from earth":PRINTTAB(29)"phase of planet";
167 QQ$=INKEY$:IFQQ$="1"THEN100:ELSEIFQQ$=CHR$(32)THENGOSUB2640:
ELSEGOTO167
168 GOSUB7025:CLS:GOTO100
170 CLS
```

Listing continues

The Key Box

Model I & III
16K RAM
Basic Level II

grees, minutes and seconds; 60 seconds equal one minute and 60 minutes equal one degree.

Longitude refers to how far around a point is on the Earth's surface. The key question is how far around from what, as there is no physical north–south circle to measure longitude from.

Greenwich, England is designated as 0 degrees longitude. Points west of this line are west longitude and points to the east are east longitude.

Longitude ranges east and west from 0–180 degrees; the 180-degree longitude line completes the circle begun by the 0-degree longitude line. Longitude is also expressed in degrees, minutes and seconds.

Declination is similar to latitude and uses the same reference point as latitude, the equator.

For example, if the Earth is a spinning ball in the center of a giant sphere and the circle formed by the Earth's equator were to expand until it touched this celestial sphere, it would trace a great celestial equator. Declination is measured in degrees north or south of this imaginary equator.

Right ascension is similar to longitude; both have an arbitrarily assigned reference point.

RA's reference point is the vernal equinox (or the first point of Aries). You don't have to locate the vernal equinox to use RA. A reference point has been established, and star charts and positions can be computed using this reference point.

Until now, RA and DEC have been almost identical to longitude and latitude. Now RA is expressed in hours, minutes and seconds instead of degrees, minutes and seconds.

The idea of measuring locations with a unit of time is slippery, but hours isn't a unit of time, it's a unit of measurement equal to 15 degrees. There are 24 hours in a circle, just as there are 360 degrees in a circle (24*15 = 360).

Right ascension is measured as you travel west from the vernal equinox. The vernal equinox is at RA Øh Øm Øs; if you travel 90 degrees to the west you're at RA 6h Øm Øs; 180 degrees to the west is RA 12h Øm Øs; 270 degrees is RA 18h Øm Øs. When you complete the circle, you're back at RA Øh Øm Øs (RA 24h = RA Øh).

Figure 1 shows RA and DEC on the star chart. Note that the celestial equator—the 0 DEC line—runs through the belt of Orion.

Now find RA 03h 43m, DEC 23°43 'N

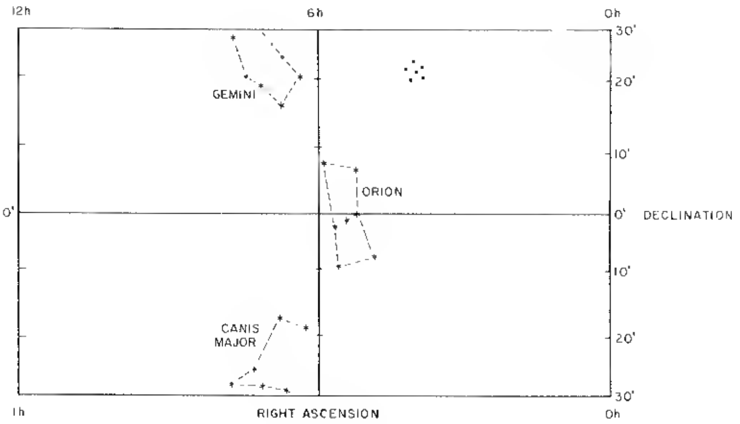


Figure 1

Listing continued

```
175 END
180 FORI=1TO300:NEXTI
185 CLS:PRINT:PRINT"  On";D;", the EARTH was located :"
187 PRINT:PRINT"DIRECTLY UNDER YOUR FEET"
188 GOSUB7025
189 CLS:GOTO100
190 CLS:PRINT@20,"STAR TRACK / SUN MENU"
195 PRINT@266,"1. Determine coordinates of SUN":PRINT@330,"2. De
termine distance and angular size":PRINT@394,"3. Determine sunri
se/sunset":PRINT@458,"4. Return to main menu"
200 PRINT@138,"Please press number of function";
210 QQ$=INKEY$:IFQQ$<"1"ORQQ$>"4"THEN210:ELSEIP=VAL(QQ$)
231 IFIP=4THENCLS:GOTO10
232 PRINT@138,"";:INPUT"Please enter the desired date (MMDDYY)";
D
233 GOSUB4000:IFNO=1THENGOTO232:ELSEID=IM
235 ONIPGOTO260,265,275
260 GOSUB2600:GOSUB3000:S$="the SUN":GOSUB3500
261 GOSUB7025:CLS:GOTO190
265 GOSUB2600:GOSUB2620:GOSUB7010:CLS:PRINT@64,"On";D;", the SUN
 will have:":PRINT:PRINT"anqular size of";IX;CHR$(130);IY;"'";IZ
"''":PRINT:PRINT"at a distance of";SR;"kilometers"
270 GOSUB7025:CLS:GOTO190
275 CLS:KH=0
276 GOSUB2630
278 TG=TR:GOSUB7020:TM=TG:GOSUB7010:I1=IX:I2=IY:I3=IZ:TG=TS:GOSU
B7020:TM=TG:GOSUB7010:I4=IX:I5=IY:I6=IZ
280 PRINT:PRINT"The SUN will rise at approximately";I1;":";I2;"a
m"
290  PRINT"    and will set  at approximately";I4;":";I5;"pm"
295 GOSUB7025:CLS:GOTO190
300 CLS:PRINT@20,"STAR TRACK / MOON MENU"
305 PRINT@266,"1. Determine coordinates of MOON":PRINT@330,"2. D
etermine distance, angular size, and phase":PRINT@394,"3. Determ
ine rise/set times":PRINT@458,"4. Return to main menu"
310 PRINT@138,"Please press number of function";
320 QQ$=INKEY$:IFQQ$<"1"ORQQ$>"4"THEN320:ELSEIP=VAL(QQ$)
321 IFIP=4THENCLS:GOTO10
322 PRINT@138,"";:INPUT"Please enter the desired date (MMDDYY)";
D
323 GOSUB4000:IFNO=1THENGOTO322:ELSEID=IM
324 IFIP=3THEN330:ELSEPRINT@176,"                    ":PRINT@138,"";
:INPUT"Please enter the desired time (HHMM)";C
325 CH!=INT(C/100):CM=C-CH*100:CM=CM/60:CH=(CH+CM)/24:ID!=ID+CH
330 ONIPGOTO340,350,375
340 GOSUB2600:GOSUB2700:GOSUB3000:S$="the MOON":GOSUB3500
345 GOSUB7025:CLS:GOTO300
350 GOSUB2600:GOSUB2700:D1=180-X6+VL:IFD1<0THEND1=D1+360:ELSEIFD
1>360THEND1=D1-360
355 F=(1+COS(D1*RA))/2:IFF>.99THENF=I.0
360 P=(1-.0549[2)/(1+.0549*COS((VM+VC)*RA)):TH=.5181/P:P=P*38440
1
361 P=INT(P):TM=TH:GOSUB7010
365 CLS:PRINT:PRINT:PRINT"The distance from earth is";P;"kilomet
ers"
367 PRINT"The angular diameter is";IX;CHR$(130);IY;"'";IZ"''":PR
INT"The phase is";F
370 GOSUB7025:CLS:GOTO300
375 CLS:PRINT:INPUT"Please enter your approximate latitude";TH:I
FTH<0ORTH>90THEN375
```

Listing continues

on the star chart. You should see a few dots north and east of Orion. If you locate Orion in the night sky, look east and north and you'll see a tiny smudge of light. If the night is dark, you might see what looks like a tiny Big Dipper. If you look through a pair of binoculars, you'll see one of the most beautiful sights in the sky, the Pleiades (play-uh-deez), or the Seven Sisters.

To explain precession and epoch, use the latitude and longitude examples again. Suppose there's an island located at 50° north latitude, 40° west longitude. Unfortunately, it's a floating island, moving one degree west every year. In 1982, its coordinates might be 50° west, 40° north; in 1983, its coordinates would be 51° west, 40° north; in 1984, 52°W 40°N; and so on. The island has a steady precession; its location or coordinates are always changing.

"...you'll see one of the most beautiful sights in the sky."

If you look at a map of the island, you want to know the year the map is good for. So, on a 1982 map, Won't-Stay-Still Island is at 50°W 40°N. The epoch of the map is 1982, which is the year that the map's coordinates are correct.

Star coordinates aren't the same every year. The Earth's axis tilts; it wobbles in its rotation. The effect is small; the Earth completes one wobble every 25,800 years. Over the years, though, the precession (change) is enough to change star coordinates.

Star charts are written for particular epochs, currently either epoch 1950 or epoch 2000. Neither is exactly correct now, but the error is slight and doesn't affect amateur observing.

Star Track uses epoch 1975 for its coordinates as a compromise between 1950 star charts and 2000 star charts. Star Track can refigure coordinates to new epochs.

Using the Program

Load and run the program; three title pages appear. Each remains on the screen for several seconds. Star Track uses this slack time to load the variables that it needs.

After the title pages, Star Track displays the master menu.

Press 1 through 6 to select any op-

Listing continued

```
377 CH=0:GOSUB2600:GOSUB2700:GOSUB3000:DB=.05*COS((VL-VN)*RA):DA
=.55+.06*COS(VM*RA):XG=XI:XH=XU:XA=XA+(12*DA):XB=XB+(12*DB):GOSU
B3000
378 GOSUB3000
379 YB=XG:YA=XH:GOSUB2631:A1=TR:A2=TS:YB=XI:YA=XU:GOSUB2631:B1=T
R:B2=TS
380 TR=(12*A1)/(12+A1-B1):TS=(12*A2)/(12+A2-B2)
381 TG=TR:GOSUB7020:TM=TG:GOSUB7010:I1=IX:I2=IY:I3=IZ:TG=TS:GOSU
B7020:TM=TG:GOSUB7010:I4=IX:I5=IY:I6=IZ
383 PRINT:PRINT"    The MOON will rise at approximately";I1;":";
I2;":";I3:    PRINT"          and will set  at approximately";I4;
":";I5;":";I6:GOSUB7025:CLS:GOTO300
400 CLS:PRINT@20,"STAR TRACK / PRECESSION & RISE/SET"
405 PRINT@266,"1. Determine precession from 1950":PRINT@330,"2.
Determine precession from 1975":PRINT@394,"3. Determine precessi
on from 2000":PRINT@458,"4. Determine rise and set times":PRINT@
522,"5. Return to main menu"
410 PRINT@138,"Please press number of function";
420 QQ$=INKEY$:IFQQ$<"1"ORQQ$>"5"THEN420:ELSEIP=VAL(QQ$)
421 IFIP=5THENCLS:GOTO10
425 IFIP=4THENGOTO460
426 IFIP=1THENE1=1950:MS=3.07327:NS=1.33617:AS=20.0426:GOTO430
427 IFIP=2THENE1=1975:MS=3.07374:NS=1.33603:AS=20.0405:GOTO430
428 E1=2000:MS=3.07420:NS=1.33589:AS=20.0383:GOTO430
430 CLS:PRINT@138,"";:INPUT"Please enter desired epoch (ex. 1981
.8)";E:IFE<1950THEN430
432 CLS:PRINT@138,"";:INPUT"Please enter RIGHT ASCENSION (HHMMSS
)";A:II=INT(A/10000):IFII<0ORII>24THEN432
434 CLS:PRINT@138,"";:INPUT"Please enter DECLINATION (DDMMSS)";B

435 IX=INT(A/10000):IY=INT((A-IX*10000)/100):IZ=INT(A-IX*10000-I
Y*100):GOSUB7000:A1=TM
437 IX=INT(B/10000):IY=INT((B-IX*10000)/100):IZ=INT(B-IX*10000-I
Y*100):GOSUB7000:B1=TM
439 AD=A1*15:SW=(MS+NS*SIN(AD*RA)*TAN(B1*RA))*(E-E1):SW=SW/3600:
TM=A1+SW:GOSUB7010:I1=IX:I2=IY:I3=IZ:S2=AS*COS(AD*RA)*(E-E1):S2=
S2/3600:TM=S2+B1:GOSUB7010:I4=IX:I5=IY:I6=IZ
441 PRINT@330,"The adjusted values for epoch";E;"are:"
443 PRINT@404,"RIGHT ASCENSION";I1;"h";I2;"m";I3;"s"
445 PRINT@468,"DECLINATION    ";I4;CHR$(131);I5;"'";I6;"''"
447 GOSUB7025:CLS:GOTO400
460 PRINT@138,"";:INPUT"Please enter the desired date (MMDDYY)";
D:GOSUB4000:IFNO=1THENGOTO232:ELSEID=IM:CLS:PRINT@138,"";:INPUT"
Please enter your approximate latitude";TH:IFTH<0ORTH>90GOTO460
461 KH=0:CLS
462 PRINT@138,"";:INPUT"Please enter RIGHT ASCENSION (HHMMSS)";A
:II=INT(A/10000):IFII<0ORII>24THEN462
463 CLS
464 PRINT@138,"";:INPUT"Please enter DECLINATION (DDMMSS)";B
465 IX=INT(A/10000):IY=INT((A-IX*10000)/100):IZ=INT(A-IX*10000-I
Y*100):GOSUB7000:A1=TM
467 IX=INT(B/10000):IY=INT((B-IX*10000)/100):IZ=INT(B-IX*10000-I
Y*100):GOSUB7000:B1=TM
468 YB=B1:YA=A1:TW=(-TAN(TH*RA)*TAN(YB*RA)):IFABS(TW)>1THENGOTO4
69:ELSEGOSUB2631:GOTO470
469 PRINT@394,"  The object either does not rise above the horiz
on,":PRINT@458," or it is circumpolar.  It does not rise and se
t.":GOTO477
470 TG=TR:GOSUB7020:TM=TG:GOSUB7010:I1=IX+KH:I2=IY:I3=IZ:TG=TS:G
OSUB7020:TM=TG:GOSUB7010:I4=IX+KH:I5=IY:I6=IZ
473 PRINT@394,"  The object will rise at";I1;":";I2;":";I3
475 PRINT@458,"          and will set  at";I4;":";I5;":";I6
477 GOSUB7025:CLS:GOTO400
500 CLS:PRINT@20,"STAR TRACK / TIME AND COORDINATES"
505 PRINT@266,"1. Convert mean time to sidereal time"
506 PRINT@330,"2. Convert sidereal time to mean time"
508 PRINT@394,"3. Return to main menu"
520 PRINT@138,"Please press number of function";
530 QQ$=INKEY$:IFQQ$<"1"ORQQ$>"3"THEN530:ELSEIP=VAL(QQ$)
532 IFIP=3THENCLS:GOTO10
550 PRINT@138,"";:INPUT"Enter the desired date  <MMDDYY>";D:GOSU
B4000:IFNO=1THENGOTO550
551 YY=D-(INT(D/100)*100):IFYY>0ANDYY<74THEN550
552 PRINT@168,"                              ":PRINT@138,"";:INPUT"Enter t
he desired time  <HHMMSS>";T
553 CH=INT(T/10000):CM=INT((T-(CH*10000))/100):CS=T-(CH*10000)-(
CM*100)
554 IFCH<0ORCH>24THEN552:ELSEIFCM>59ORCM<0THEN552:ELSEIFCM>59ORC
M<0THEN552
555 IX=CH:IY=CM:IZ=CS:GOSUB7000:IFYY=00THENYY=25:ELSEYY=YY-75
556 ONIPGOTO560,557
557 UT=UA*KM-UB(YY):IFUT<0THENUT=UT+24
558 TM=TM-UT:IFTM<0THENTM=(TM+24)*UD:GOTO562:ELSETM=TM*UD:GOTO56
2
```

Listing continues

✓See List of Advertisers on Page 563

Listing continued

```
560 UT=UA*KM-UB(YY):TM=TM*UC+UT:IFTM>24THENTM=TM-24:ELSEIFTM<ØTH
ENTM=TM+24
562 GOSUB7Ø1Ø:IFIP=1THENGOTO564:ELSECLS:PRINT:PRINT"The mean tim
e is";IX;":";IY;":";IZ:GOSUB7Ø25:CLS:GOTO5ØØ
564 CLS:PRINT:PRINT"The sidereal time is";IX;":";IY;":";IZ:GOSUB
7Ø25:CLS:GOTO5ØØ
22ØØ OP=(36Ø/365.25)*(ID/T1(IP)):OP=OP-(INT(OP/36Ø)*36Ø)
22Ø5 X1=OP+(36Ø/3.1415927)*T4(IP)*SIN((OP+T2(IP)-T3(IP))*RA)+T2(
IP)
221Ø IFX1>36ØTHENX1=X1-36Ø:ELSEIFX1<ØTHENX1=X1+36Ø
2215 VP=X1-T3(IP)
222Ø XR=(T5(IP)*(1-T4(IP)[2))/(1+T4(IP)*COS((VP*RA)))
2225 X2=SIN((X1-T7(IP))*RA)*SIN(T6(IP)*RA)
2227 X2=ATN(X2/SQR(-X2*X2+1))
223Ø X2=X2*57.29578
2235 X3=ATN(TAN((X1-T7(IP))*RA)*COS(T6(IP)*RA))*57.29578+T7(IP)
2236 GOSUB235Ø
2237 X4=XR*COS(X2*RA)
224Ø RETURN
23ØØ PN=(36Ø/365.25)*(ID/T1(2))
23Ø5 PN=PN-(INT(PN/36Ø)*36Ø)
231Ø PL=PN+(36Ø/3.1415927)*T4(2)*SIN((PN+T2(2)-T3(2))*RA)+T2(2)
2315 IFPL>36ØTHENPL=PL-36Ø:ELSEIFPL<ØTHENPL=PL+36Ø
232Ø PV=PL-T3(2)
2325 PR=(1-T4(2)[2)/(1+T4(2)*COS(PV*RA))
234Ø RETURN
235Ø XZ=X1*.Ø5
236Ø FORJJ=-36ØTO36ØSTEP18Ø
2365 XQ=X3+JJ
237Ø IFABS(XQ-X1)<XZTHENX3=XQ:RETURN
2375 NEXTJJ
238Ø PRINT"ERROR":END
24ØØ XA=ATN((X4*SIN((PL-X3)*RA))/(PR-X4*COS((PL-X3)*RA)))
241Ø XA=(XA*57.29578)+PL+18Ø
2415 IFXA>36ØTHENXA=XA-36Ø:ELSEIFXA<ØTHENXA=XA+36Ø
242Ø XB=ATN((X4*TAN(X2*RA)*SIN((XA-X3)*RA))/(PR*SIN((X3-PL)*RA))
)
2425 XB=XB*57.29578
243Ø RETURN
25ØØ XA=ATN((PR*SIN((X3-PL)*RA))/(X4-PR*COS((X3-PL)*RA)))
25Ø5 XA=(XA*57.29578)+X3
251Ø IFXA>36ØTHENXA=XA-36Ø:ELSEIFXA<ØTHENXA=XA+36Ø
2515 XB=(ATN((X4*TAN(X2*RA)*SIN((XA-X3)*RA))/(PR*SIN((X3-PL)*RA)
)))
252Ø XB=XB*57.29578
253Ø RETURN
26ØØ SN=(36Ø/365.25)*ID:SN=SN-(INT(SN/36Ø)*36Ø):SM=SN+279.Ø4147Ø
-282.51Ø396:IFSM<ØTHENSM=SM+36Ø
26Ø5 SE=(36Ø/3.1415927)*.Ø1672*SIN(SM*RA):XA=SN+SE+279.Ø4147:IFX
A>36ØTHENXA=XA-36Ø
261Ø XB=Ø:RETURN
262Ø SV=SM+SE:SF=(1+(.Ø1672*COS(SV*RA)))/(1-.Ø1672[2):SR=1495958
5ØØ/SF:SØ=SF*.533128:TM=SØ:SR=INT(SR):RETURN
263Ø CLS:INPUT"Please enter your approximate latitude";TH:GOSUB2
6ØØ:GOSUB3ØØØ:IX=I1+KH:IY=I2:IZ=I3:GOSUB7ØØØ:YA=TM:IX=I4+KH:IY=I
5:IZ=I6:GOSUB7ØØØ:YB=TM
2631 TW=(-TAN(TH*RA)*TAN(YB*RA)):TW=-ATN(TW/SQR(-TW*TW+1))+1.57Ø
8:TW=TW*57.29578/15
2632 TR=24+YA-TW:IFTR>24THENTR=TR-24
2633 TS=YA+TW:IFTS>24THENTS=TS-24
2634 RETURN
264Ø P2=PR[2+XR[2-(2*PR*XR*COS((X1-PL)*RA)):AU=SQR(P2):AT=T8(IP)
/AU:WD=XA-X1:AF=(1+COS(WD*RA))/2:IFAF>.99THENAF=1.Ø
2642 PRINT@448,"The distance from earth is";AU;"AU   ":PRINT@512
,"The angular diameter is";AT;"''":PRINT@576,"The phase is";AF:R
ETURN
27ØØ VL=(36Ø/27.3217)*ID!+124.8756:VL=VL-(INT(VL/36Ø)*36Ø):VM=VL
-(36Ø/365.25)*(ID!/8.85)-145.96Ø1:VM=VM-(INT(VM/36Ø)*36Ø):VN=248
.6441-(36Ø/365.25)*(ID!/18.61):VN=VN-(INT(VN/36Ø)*36Ø)
27Ø1 X6=XA
27Ø5 VE=1.274*SIN((2*(VL-XA)-VM)*RA):VA=Ø.186*SIN(SM*RA):V3=Ø.37
*SIN(SM*RA):VM=VM+VE-VA-V3:VC=6.289*SIN(VM*RA):VL=VL+VE-VA-VC
271Ø VV=.658*SIN(2*(VL-XA)*RA):VL=VL+VV:VN=VN-.16*SIN(SM*RA):XA=
ATN((TAN((VL-VN)*RA)*COS(5.1453*RA)))*57.29578:XA=XA+VN:X1=VL:X3
=XA:GOSUB235Ø:XA=X3
2715 XB=SIN((VL-VN)*RA)*SIN(5.1453*RA):XB=ATN(XB/SQR(-XB*XB+1))*
57.29578
272Ø RETURN
3ØØØ XB=XB*RA:XA=XA*RA
3ØØ5 XT=SIN(XB)*COS(RE)+COS(XB)*SIN(RE)*SIN(XA)
3Ø1Ø XT=ATN(XT/SQR(-XT*XT+1))
3Ø11 XT=XT*57.29578
3Ø15 X9=(TAN(XA)*COS(RE))-((TAN(XB)*SIN(RE))/COS(XA))
3Ø2Ø X9=ATN(X9)*57.29578:XA=XA/RA
```

Listing continues

tion. The menu for option 1 looks like this:

Star Track/Planet Tracker
Please select an option
1. Mercury
2. Venus
3. Earth
4. Mars
5. Jupiter
6. Saturn
7. Uranus
8. Neptune
9. Pluto
A. Return to main menu

Press 1 through 9, or A to return to menu. If you select a planet, you're asked to enter a date. Enter the date in MMDDYY format and press enter; you can use any date from 1950 to 2000.

Star Track then displays the RA and DEC of your planet. It asks you to press 1 to see the angular size, phase and distance from Earth in AU or to press the space bar to return to the last menu.

The menu for option 2 looks like this:

Star Track/Sun Menu
Please press number of function
1. Determine coordinates of sun
2. Determine distance and angular size
3. Determine sunrise/sunset
4. Return to main menu

If you select option 3 (sunrise and sunset) you are prompted for your approximate latitude. If you live north of the equator, enter a positive value; if you live south of the equator, enter a negative value.

The menu for option 3 looks like this:

Star Track/Moon Menu
Please press number of function
1. Determine coordinates of moon
2. Determine distance, angular size and phase
3. Determine rise/set times
4. Return to main menu

Since the moon has such a high apparent motion, you're prompted for the time as well as the date. Enter the time in an HHMM format using military time (i.e., 1800 = 6 p.m.).

For rise/set times, you're not asked to enter the time, but you must enter your approximate latitude.

The menu for option 4 is below:

Star Track/Precession and Rise/Set
Please press number of function
1. Determine precession from 1950
2. Determine precession from 1975
3. Determine precession from 2000
4. Determine rise and set times
5. Return to main menu

If you select 1, 2 or 3, Star Track prompts you for the epoch you wish to convert to. If you select function 1, you have a set of 1950 coordinates to convert to a new epoch.

This epoch is a four-digit year with an additional decimal digit denoting parts of a year for a total of five digits. For example, June 1979 is halfway through 1979. Six months is .5 of a year, so the proper epoch for June 1979 is 1979.5. October of 1986 is 10 months into 1986. 10 months is about 8/10 of a year, so the epoch is 1986.8.

"Star coordinates aren't the same every year."

You can enter any epoch between 1975 and 2000. Once you enter the epoch you want to convert to, you're asked for right ascension and declination. Enter RA in HHMMSS format and enter DEC in DDMMSS format. Declinations north of the celestial equator are positive; those south of the equator are negative.

For example:

(RA = 12h 18m 03s) Please enter right ascension (HHMMSS)? 121803. (DEC = 16°S, 14′ 08″) Please enter declination (DDMMSS)? −161408.

If you select function 4, you're prompted for the date, your approximate latitude, and the RA and DEC of the desired object.

Option 5's menu looks like this:

```
Star Track/Time
Please press number of function
1. Convert mean time to sidereal time
2. Convert sidereal time to mean time
3. Return to main menu
```

Functions 1 and 2 prompt you for the time and the date.

Hints

- All rise and set computations require your approximate latitude.
- Daylight-saving time is not used.
- When you respond to a request for a date, the program remembers that date. Pressing enter to the next prompt for a date results in the original date being used. This is handy when you want to determine several different functions for the same date, for example, positions of the planets, and rise/set of the sun and moon. ■

Listing continued

```
3021 IFXA<=90THENIQ=1:ELSEIFXA<=180THENIQ=2:ELSEIFXA<=270THENIQ=
3:ELSEIQ=4
3022 IFX9<0THENX9=X9+90:GOTO3022
3023 IFX9>360THENX9=X9-90:GOTO3023
3024 IFX9<=90THENJQ=1:ELSEIFX9<=180THENJQ=2:ELSEIFX9<=270THENJQ=
3:ELSEJQ=4
3025 X9=X9+((IQ-JQ)*90)
3029 X9=X9/15
3030 IFX9<0THENX9=X9+24
3031 XU=X9:XI=XT
3035 I1=INT(X9):X9=X9-I1
3040 I2=INT(X9*60):X9=(X9*60)-I2
3045 I3=INT(X9*60)
3050 I4=INT(XT):XT=XT-I4
3055 I5=INT(XT*60):XT=(XT*60)-I5
3060 I6=INT(XT*60)
3065 RETURN
3500 CLS
3510 PRINT@128,"  On";D;", ";S$;" will be at : "
3515 PRINT@256,"RIGHT ASCENSION = ";I1;"h";I2;"m";I3;"s"
3520 PRINT@320,"DECLINATION     = ";I4;CHR$(130);I5;"'";I6;"''"
3540 RETURN
4000 IM=INT(D/10000)
4005 ID=INT((D-IM*10000)/100)
4010 IY=(D-(IM*10000)-(ID*100)):KY=IY
4011 IFIM<1ORIM>12THENNO=1:RETURN:ELSEIFID<1ORID>31THENNO=1:RETU
RN
4012 NO=0
4015 IL=INT(IY/4)*4
4020 IFIL=0THENLL=0:ELSEIFIL=IYTHENLL=1:ELSELL=0
4025 IFIM>2THEN4040
4030 IM=(IM-1)*(63-LL):IM=IM/2
4035 GOTO4050
4040 IM=INT((IM+1)*30.6)
4045 IM=IM-63-LL
4050 IM=IM+ID
4053 IFIY<75THENIY=IY+100
4054 KM=IM
4055 IY=IY-75:IFIY=0THENRETURN:ELSEIH=1
4057 KM=IM
4060 FORI=1TOIY
4065 IFI=IHTHENIM=IM+366:IH=IH+4:ELSEIM=IM+365
4070 IFIH=25ANDI=IHTHENIM=IM-1
4075 NEXTI
4085 RETURN
4100 DATA"MERCURY",.24085,320.66305,77.06645,.205629,.387099,7.0
0427,48.03493,6.74,1.918E-6
4105 DATA"VENUS",.61521,310.97453,131.21928,.006785,.723332,3.39
428,76.45475,16.92,1.721E-5
4110 DATA"EARTH",1.00004,99.53431,102.51044,.016720,1,0,0,0,0
4115 DATA"MARS",1.88089,249.62919,335.59881,.093382,1.523691,1.8
4983,49.36466,9.36,4.539E-6
4120 DATA"JUPITER",11.86224,355.21414,13.91992,.04846,5.202804,1
.3045,100.19608,196.74,1.994E-4
4125 DATA"SATURN",29.45771,104.17278,92.55833,.05563,9.538844,2.
48933,113.43842,165.6,1.74E-4
4130 DATA"URANUS",84.01247,205.78286,170.25472,.04725,19.181854,
.77316,73.87283,65.8,7.768E-5
4135 DATA"NEPTUNE",164.79558,249.91462,44.40592,.008586,30.05796
,1.77236,131.50506,62.2,7.597E-5
4140 DATA"PLUTO",246.378,202.3345,224.2580,.246115,39.29976,17.1
4451,109.9965,8.20,4.073E-6
4150 DATA.397221,.413525,.363611,.379644,.395588,.411473,.361678
,.377595,.393506,.409421,.359625,.37554,.391454,.407368,.357573,
.373487,.389402,.405316,.355521,.371435,.387349,.403264,.353468,
.369383,.385297,.401211
7000 TM=(((IZ/60)+IY)/60)+IX:RETURN
7010 IX=INT(TM):TM=TM-IX:IY=INT(TM*60):TM=(TM*60)-IY:IZ=INT(TM*6
0):RETURN
7020 W1=KM*UA-UB(KY-75):IFW1<0THENW1=W1+24
7022 TG=TG-W1:IFTG<0THENTG=(TG+24)*UD:ELSETG=TG*UD
7024 RETURN
7025 PRINT@900,"Press <SPACE BAR> to return to last menu"
7027 QQ$=INKEY$:IFQQ$<>CHR$(32)THEN7027:ELSERETURN
7050 PRINTCHR$(23)
7055 PRINT@278,"STAR TRACK"
7060 FORI=0TO2:READP$(I):READT1(I):READT2(I):READT3(I):READT4(I)
:READT5(I):READT6(I):READT7(I):READT8(I):READT9(I):NEXTI
7065 PRINT@398,"AN ASTRONOMY GUIDE"
7070 FORI=3TO8:READP$(I):READT1(I):READT2(I):READT3(I):READT4(I)
:READT5(I):READT6(I):READT7(I):READT8(I):READT9(I):NEXTI
7075 CLS:PRINT@281,"An aid for the":PRINT@407,"amateur astronome
r":PRINT@539,"written by:":PRINT@667,"Joey Robichaux"
7080 FORI=0TO25:READUB(I):UB(I)=UB(I)+17:NEXTI
7085 RETURN
```

LOAD 80

Datascope

by Dennis Ridgway

Like an oscilloscope for software, Datascope reads data bit by bit, letting you recover data from glitched tapes or see data on tape.

Have you ever needed to retrieve a key piece of data from an unreadable tape, or wondered about the speed of your cassette recorder, wanted to know the format of some cassette file, or wished you could visually inspect the recorded image on a tape? This Level II 16K analyzer program addresses these problems.

This program is sort of a software oscilloscope for viewing pulses and gaps, those most fundamental particles of recorded data. Or it might be called a data microscope, allowing you to focus on recorded data at different levels of magnification, ranging from pulses and gaps, through bits and bytes, to Basic token words and two-byte pair values.

The Program

The program performs four distinct operations. The first, the load operation, reads data from a tape and stores it in an area of memory called the buffer. The next three operations work with the stored data. The analyzer operation observes the pulse and gap patterns to determine the clock pulse alignment necessary for data bit interpretation. The search operation determines the bit alignment necessary for data byte interpretation. The translate operation changes data bytes into meaningful data forms.

The program is written in Basic and uses two Assembly-language subroutines. One subroutine is used in the load operation to read data from the cassette; the other is used by the search and translate operations for converting pulse and gap data into logical bits.

Load Operation

This operation loads data from any cassette tape into a 7000-byte buffer. It is not a read routine as no logical interpretation of the data is made, but is a transfer of pulse and gap sensings. The sensings, called peeps, are taken at precise time intervals. Each peep detects whether the read head of the recorder is positioned over a pulse or a gap and the result is stored in the buffer as a one bit or a zero bit, respectively.

The time interval between peeps is operator selectable and can range from 58.5 microseconds to more than 1700 microseconds. Since both pulses and gaps usually last longer than 100 microseconds (at the 500-baud rate), multiple peeps are possible during the span of a single pulse or gap. For the interpreting operations to work properly, multiple peeps are desirable. Normally, intervals greater than 143 microseconds produce data that cannot be interpreted reliably by the other routines and intervals greater than 214 microseconds produce unintelligible data; whole pulses or gaps may be bridged between peeps.

Because the peeps are at precise time intervals, it is possible to make observations regarding your cassette player's running speed, acceleration time, and pulse width to gap width relationships for different volume settings. This information can be useful in diagnosing problems. The load operation does not perform an interpretive read like your familiar load commands; it can, therefore, be used to

The Key Box

Model I
16K RAM
Level II

load data from any tape and from any point on a tape. This makes it useful for identifying unlabeled tapes and files and for recovering partially overwritten files.

Analyze Operation

This operation performs the initial steps of interpreting the raw data captured by the load operation. The operator is allowed to view the peeps in two different formats.

The first format, shown in Fig. 1, presents the peeps in a straight vertical line. Peeps are printed as # for pulses and — for gaps. A waveform and notations have been added to Fig. 1 to visualize pulse and gap patterns.

"You can view the peeps in two different formats."

In the second format, shown in Fig. 2, the same peeps are being displayed. Here they have been separated into groups, with a group on each line representing one logical bit of information. Each print line starts with # peeps which it considers to be a clock pulse. These are followed by — peeps (a gap). Another series of # peeps indicates this is a one bit or a string of — peeps continue indicating this is a zero bit.

This vertical alignment of clock pulses is accomplished by the operator trying different bit widths during the analyze operation. The bit width is the average number of peeps from the start of one clock pulse to the start of the next. The routine starts at a clock pulse and skips across a number of peeps equivalent to .874 of the test bit width. At this time, the routine starts looking for the next pulse. When one is found it is considered to be the next clock pulse and a new line is started.

The T16 on the first line of Fig. 2 indicates the test bit width was 16. The whole number result of multiplying 16 by .874 is 13; the search for the next clock pulse was started after 13 peeps were printed.

Each line indicates the buffer address AD of the first peep on that line. Notice the print line with address AD2213:6 in Fig. 2. This shows that peep six of byte number 2213 was the location of the first peep on this line. (Buffer addresses run from 1–7000 and peep numbers run from 1–8.)

The T16 shows the test bit width and

```
AD 2213 :1  T O _      1
AD 2213 :2  T O _      1
AD 2213 :3  T O _      1
AD 2213 :4  T O _      1
AD 2213 :5  T O _      1
AD 2213 :6  T O #      1
AD 2213 :7  T O #      1
AD 2213 :8  T O #      1
AD 2214 :1  T O #      1
AD 2214 :2  T O _      1
AD 2214 :3  T O _      1
AD 2214 :4  T O _      1
AD 2214 :5  T O _      1
AD 2214 :6  T O #      1
AD 2214 :7  T O #      1
AD 2214 :8  T O #      1
AD 2215 :1  T O #      1
AD 2215 :2  T O _      1
AD 2215 :3  T O _      1
AD 2215 :4  T O _      1
AD 2215 :5  T O #      1
AD 2215 :6  T O #      1
AD 2215 :7  T O #      1
AD 2215 :8  T O #      1
AD 2216 :1  T O _      1
AD 2216 :2  T O _      1
AD 2216 :3  T O _      1
AD 2216 :4  T O _      1
AD 2216 :5  T O _      1
AD 2216 :6  T O _      1
AD 2216 :7  T O _      1
AD 2216 :8  T O _      1
AD 2217 :1  T O _      1
AD 2217 :2  T O _      1
AD 2217 :3  T O _      1
AD 2217 :4  T O _      1
AD 2217 :5  T O #      1
AD 2217 :6  T O #      1
AD 2217 :7  T O #      1
AD 2217 :8  T O #      1
AD 2218 :1  T O _      1
AD 2218 :2  T O _      1
AD 2218 :3  T O _      1
AD 2218 :4  T O _      1
AD 2218 :5  T O #      1
AD 2218 :6  T O #      1
AD 2218 :7  T O #      1
AD 2218 :8  T O #      1
AD 2219 :1  T O _      1
AD 2219 :2  T O _      1
AD 2219 :3  T O _      1
AD 2219 :4  T O #      1
AD 2219 :5  T O #      1
AD 2219 :6  T O #      1
AD 2219 :7  T O #      1
AD 2219 :8  T O _      1
AD 2220 :1  T O _      1
AD 2220 :2  T O _      1
AD 2220 :3  T O _      1
AD 2220 :4  T O _      1
AD 2220 :5  T O _      1
```

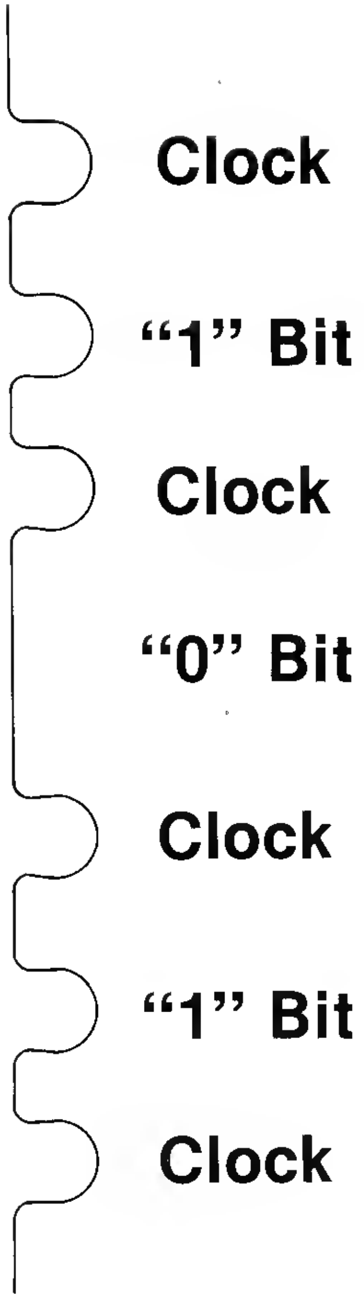


Figure 1

the 15 at the right shows the actual bit width of this bit line. The pulse/gap pattern of #### ————####——— shows this to be a one bit. You may notice this same pattern in Fig. 1, running vertically from line AD2213:6 to AD2215:4. This one bit happens to be the first one bit of the synchronization bit pattern of 10100101 at the beginning of data on a tape file.

While the analyze operation was running, the right-arrow key (a delta in Fig. 2) and the left-arrow key (printed as a right bracket) increase or decrease the test bit width by one. Note line AD2270:8 of Fig. 2. The delta shows that a right arrow was invoked increasing the T value from 16 to 17. The right

```
AD 2203 :8     T 16 ####___________             15
AD 2205 :7     T 16 ####____________            16
AD 2207 :7     T 16 ####___________             15
AD 2209 :6     T 16 ####____________            16
AD 2211 :6     T 16 ####____________            16
AD 2213 :6     T 16 ####____####___             15
AD 2215 :5     T 16 ####____________            16
AD 2217 :5     T 16 ####____####___             15
AD 2219 :4     T 16 ####____________            16
AD 2221 :4     T 16 ####___________             15
AD 2223 :3     T 16 ####____####____            16
AD 2225 :3     T 16 ####___________             15
AD 2227 :2     T 16 ####____####_____________   25
AD 2230 :3     T 16 ####___####____             15
AD 2232 :2     T 16 ####____####___             15
AD 2234 :1     T 16 #####___________            16
AD 2236 :1     T 16 ####____####___             15
AD 2237 :8     T 16 #####___________            16
AD 2239 :8     T 16 ####___________             15
AD 2241 :7     T 16 ####____####____            16
AD 2243 :7     T 16 ####___#####____            16
AD 2245 :7     T 16 ####___#####___             15
AD 2247 :6     T 16 ####____####___             15
AD 2249 :5     T 16 #####___________            16
AD 2251 :5     T 16 ####____####___             15
AD 2253 :4     T 16 ####____________            16
AD 2255 :4     T 16 ####___________             15
AD 2257 :3     T 16 ####____####____            16
AD 2259 :3     T 16 ####___#####____            16
AD 2261 :3     T 16 ####___####____             15
AD 2263 :2     T 16 ####____####___             15
AD 2265 :1     T 16 ####____________            16
AD 2267 :1     T 16 ####____####___             15
AD 2268 :8     T 16 ####____________            16
AD 2270 :8 ^   T 17 ####___________             15
AD 2272 :7     T 17 ####____####____            16
AD 2274 :7     T 17 ####___#####____            16
AD 2276 :7     T 17 ####___________             15
AD 2278 :6 ]   T 16 ####____####____            16
AD 2280 :6     T 16 ####___________             15
AD 2282 :5     T 16 ####____________            16
AD 2284 :5     T 16 ####___________             15
AD 2386 :4 \   T 16 #_________####____________  26
AD 2389 :6     T 16 ####___####____             15
AD 2391 :5     T 16 ####____####___             15
AD 2393 :4     T 16 #####___________            16
AD 2295 :4 [   T 16 #_____####____________      22
AD 2298 :2     T 16 ####____####___             15
AD 2300 :1     T 16 ####____####____            16
AD 2302 :1     T 16 ####___#####___             15
AD 2303 :8     T 16 ####____________            16
AD 2305 :8     T 16 ####____________            16
AD 2307 :8     T 16 ####___________             15
AD 2309 :7     T 16 ####____####____            16
AD 2311 :7     T 16 ####___________             15
AD 2313 :6     T 16 ####____________            16
```

Figure 2

Figure 3

```
KEY 165
SEARCH ^STARTED AT BYTE- 2209 PEEP#- 6 WIDTH- 16
0010 1001   0111 0100   1111 0100   1111 0100   1101 0000   0100 0111
   29H         74H         F4H         F4H         D0H         47H
   41          116         244         244         208         71
   )           t           STR#        STR#        /           G
         29737       -2956       -2828       -12044      18384
*************************************************************************
KEY 165
```

Figure 3 continues

See List of Advertisers on Page 563

Figure 3 continued

```
SEARCH ^STARTED AT BYTE- 2211 PEEP#- 6 WIDTH- 16
0101 0010   1110 1001   1110 1001   1110 1001   1010 0000   1000 1110
   52H         E9H         E9H         E9H         A0H         8EH
   82          233         233         233         160         142
  R           EOF         EOF         EOF         OUT         RUN
       -5806       -5655       -5655      -24343      -29024
**********************************************************************
KEY 165
SEARCH ]STARTED AT BYTE- 2209 PEEP#- 6 WIDTH- 16
0010 1001   0111 0100   1111 0100   1111 0100   1101 0000   0100 0111
   29H         74H         F4H         F4H         D0H         47H
   41          116         244         244         208         71
  )           t           STR$        STR$        /           G
        29737       -2956       -2828      -12044       18384
**********************************************************************
KEY 165
SEARCH ]STARTED AT BYTE- 2207 PEEP#- 7 WIDTH- 16
0001 0100   1011 1010   0111 1010   0111 1010   0110 1000   0010 0011
   14H         BAH         7AH         7AH         68H         23H
   20          186         122         122         104         35
              CSAVE       z           z           h           #
       -17900       31418       31354       26746        9064
**********************************************************************
KEY 165
SEARCH ^STARTED AT BYTE- 2209 PEEP#- 6 WIDTH- 16
0010 1001   0111 0100   1111 0100   1111 0100   1101 0000   0100 0111
   29H         74H         F4H         F4H         D0H         47H
   41          116         244         244         208         71
  )           t           STR$        STR$        /           G
        29737       -2956       -2828      -12044       18384
**********************************************************************
KEY 165
SEARCH ^STARTED AT BYTE- 2211 PEEP#- 6 WIDTH- 16
0101 0010   1110 1001   1110 1001   1110 1001   1010 0000   1000 1110
   52H         E9H         E9H         E9H         A0H         8EH
   82          233         233         233         160         142
  R           EOF         EOF         EOF         OUT         RUN
       -5806       -5655       -5655      -24343      -29024
**********************************************************************
KEY 165
SEARCH ^STARTED AT BYTE- 2213 PEEP#- 6 WIDTH- 16
1010 0101   1101 0011   1101 0011   1101 0011   0100 0001   0001 1100
   A5H         D3H         D3H         D3H         41H         1CH
   165         211         211         211         65          28
  PUT         OR          OR          OR          A
       -11355      -11309      -11309       16851        7233
**********************************************************************
MATCH
```

```
KEY 80         69          69          80          69          82
SEARCH ^STARTED AT BYTE- 575 PEEP#- 2 WIDTH- 16
1010 1010   0000 1000   1010 1000   1010 1010   0000 1000   1010 1010
   AAH         08H         A8H         AAH         08H         AAH
   170         8           168         170         8           170
  KILL                    MERGE       KILL                    KILL
        2218       -22520      -21848        2218      -22008
**********************************************************************
KEY 80         69          69          80          69          82
SEARCH ^STARTED AT BYTE- 576 PEEP#- 7 WIDTH- 16
0101 0100   0001 0001   0101 0001   0101 0100   0001 0001   0101 0100
   54H         11H         51H         54H         11H         54H
   84          17          81          84          17          84
  T                       Q           T                       T
        4436        20753       21585        4436       21521
**********************************************************************
KEY 80         69          69          80          69          82
SEARCH ^STARTED AT BYTE- 579 PEEP#- 2 WIDTH- 16
1010 1000   0010 0010   1010 0010   1010 1000   0010 0010   1010 1001
   A8H         22H         A2H         A8H         22H         A9H
   168         34          162         168         34          169
  MERGE       "           OPEN        MERGE       "           NAME
        8872       -24030      -22366        8872      -22238
**********************************************************************
KEY 80         69          69          80          69          82
SEARCH ^STARTED AT BYTE- 581 PEEP#- 2 WIDTH- 16
0101 0000   0100 0101   0100 0101   0101 0000   0100 0101   0101 0010
   50H         45H         45H         50H         45H         52H
   80          69          69          80          69          82
  P           E           E           P           E           R
        17744       17733       20549       17744       21061
**********************************************************************
MATCH
```

Figure 4

bracket on line AD2278:6 indicates the left arrow was invoked to decrement the T value back to 16.

The down and up-arrow keys increase and decrease the buffer byte address by 100. Line AD2386:4 shows the down-arrow key (printed as backward slash) was used to increase the byte address by 100. Line AD2295:4 indicates the up-arrow key (left bracket) was invoked. Any time the up or down arrow is used, the clock pulse may be thrown out of proper alignment for a few lines. (The arrows display properly on the CRT and may print properly on your printer.)

Any other keys cause the analysis to stop, at which time you can go back to the main menu to select another operation or continue the analysis.

Search Operation

The bit width determined in the analyze operation allows the raw data to be further interpreted by this search operation. This operation scans across the bits to determine byte alignment. A frame 48 bits wide is passed over the data. The 48 bits in the current frame position are treated as six bytes and are converted into hexadecimal, decimal, character, and Basic token word forms; pairs of bytes are converted into 16-bit binary values.

The frame is then shifted one logical bit to the right (or left) and the 48 bits found in the frame are converted. This is repeated until stopped by the operator or the data is exhausted. The left and right-arrow keys change the direction of the shift. Normally, when recognizable bytes are found the operator would stop the operation, note the byte alignment address, and proceed to the translate operation. When a left-shifting search is used, bits followed by exceptionally long gaps will not be interpreted correctly. Usually two bits will be wrong before alignment is restored.

A search key, one to six bytes long, can be entered for automatically stopping the search when a match is found. Figure 3 is an example of a one-byte search key, with decimal value 165, used to stop the search when the synchronization byte was found. Notice the address of 2213:6 near the bottom. This is the same data we were observing on Figs. 1 and 2. Figure 4 is an example of a six-byte search key of 80, 69, 69, 80, 69, and 82, which was used to find the name Peeper on the Assembler-language program tape. All search keys are entered in decimal form.

If this utility program is used to

identify an unlabeled file, the search operation might be as far as you need to go. However, if you are trying to recover some otherwise unreadable data, want to convert a short Assembly routine into decimal for string packing, or want to study the format of some file, note the byte and bit address where alignment occurred and proceed to the translate operation.

Translate Operation

The output of the translate operation (Figs. 5 and 6) is the same format as the output of the search operation. Again, a frame 48 bits wide is passed over the data. In the translate mode, after 48 bits are translated, the frame is shifted to the next 48 bits to the right. The translate operation runs until stopped by the operator or the data is exhausted.

Search and Translate Output Format

The output of both the search and translate operations consists of six lines of information for each 48 bits interpreted. Line 1 names the operation (search or translate); shows the direction of the frame shift, if in the search mode (again, right arrows appear as deltas and left arrows as right brackets); shows the byte and peep address of where this line group started; and finally, shows what bit width was being used.

Line 2 shows the 48 bits as zeros and ones. Line 3 contains the hexadecimal values; line 4 shows the decimal values; and line 5 contains the characters or Basic token words. Line 6 has the 16-bit binary values derived by treating the paired bytes in the sequence of least significant byte followed by most significant byte. When operating in the translate mode, the first byte of each line group is paired with the last byte of the preceding line group.

Figure 5 (as well as Fig. 4) was prepared using a cassette that contained a copy of the Assembly object program in Listing 2. The first line group divulges the program name of Peeper. The second line group divulges the hexadecimal 3C program section identifier, the decimal 128 (the number of bytes in the program section), the 16-bit binary value of "32512" (the ORG memory address for the program), and finally the hex 1158 (the first two bytes of the program code). The remaining lines on this figure show a continuation of the hexadecimal object code. The decimal values right below each hexadecimal value could be used for string packing in a Basic program.

Figure 6 (also Figs. 1, 2, and 3) was prepared using a cassette containing a copy of the Basic Program Listing 1. The fifth line of the first line group on Fig. 6 shows "5 = VARPTR(E5" from line 50 of the program. The remainder of line 50 can be read in line groups 2 and 3 of Fig. 6. In line group 3 following the (E6) we see the decimal value zero, which is the indicator for the end of a program line. Line group 4 starts with the 16-bit binary program

```
TRANSLATE STARTED AT BYTE- 581 PEEP#- 2 WIDTH- 16
0101 0000  0100 0101  0100 0101  0101 0000  0100 0101  0101 0010
   50H        45H        45H        50H        45H        52H
   80         69         69         80         69         82
  P          E          E          P          E          R
 20562   17744      17733      20549      17744      21061
*****************************************************************
TRANSLATE STARTED AT BYTE- 673 PEEP#- 8 WIDTH- 16
0011 1100  1000 0000  0000 0000  0111 1111  0001 0001  0101 1000
   3CH        80H        00H        7FH        11H        58H
   60         128        0          127        17         88
  <          END                                         X
 15442  -32708       128        32512      4479       22545
*****************************************************************
TRANSLATE STARTED AT BYTE- 1584 PEEP#- 3 WIDTH- 16
0001 1011  0010 0001  1010 1000  0110 0011  0011 1110  0000 0001
   1BH        21H        A8H        63H        3EH        01H
   27         33         168        99         62         1
             !          MERGE      c          >
 7000      8475     -22495      25512      15971      318
*****************************************************************
TRANSLATE STARTED AT BYTE- 1677 PEEP#- 6 WIDTH- 16
0111 0111  0010 0011  0001 1011  0111 1010  1011 0011  0010 0000
   77H        23H        1BH        7AH        B3H        20H
   119        35         27         122        179        32
  w          #                     z          CONT
 30465     9079       6947       31259     -19590      8371
*****************************************************************
TRANSLATE STARTED AT BYTE- 1771 PEEP#- 2 WIDTH- 16
1111 0111  0010 0001  1010 1000  0110 0011  0001 0001  0101 1000
   F7H        21H        A8H        63H        11H        58H
   247        33         168        99         17         88
 . CHR$      !          MERGE      c                     X
-2272      8695      -22495      25512      4451       22545
*****************************************************************
TRANSLATE STARTED AT BYTE- 1864 PEEP#- 5 WIDTH- 16
0001 1011  0011 1110  0000 0100  1101 0011  1111 1111  1101 1011
   1BH        3EH        04H        D3H        FFH        DBH
   27         62         4          211        255        219
             >                     OR                    INP
 7000      15899      1086      -11516     -45        -9217
*****************************************************************
TRANSLATE STARTED AT BYTE- 1957 PEEP#- 8 WIDTH- 16
1111 1111  0001 0111  0011 0000  1111 1011  0111 1110  0001 0111
   FFH        17H        30H        FBH        7EH        17H
   255        23         48         251        126        23
                        0                     ~
-37        6143       12311     -1232       32507      6014
*****************************************************************
```

Figure 5

Figure 6

```
TRANSLATE STARTED AT BYTE- 6325 PEEP#- 8 WIDTH- 16
0011 0101  1101 0101  1100 0000  0010 1000  0100 0101  0011 0101
   35H        D5H        C0H        28H        45H        35H
   53         213        192        40         69         53
  5          =          VARPTR     (          E          5
 13823   -10955     -16171      10432      17704      13637
*****************************************************************
TRANSLATE STARTED AT BYTE- 6419 PEEP#- 5 WIDTH- 16
0010 1001  0011 1010  0100 0110  0011 0110  1101 0101  1100 0000
   29H        3AH        46H        36H        D5H        C0H
   41         58         70         54         213        192
  )          :          F          6          =          VARPTR
 10549     14889      17978      13894     -10954     -16171
*****************************************************************
TRANSLATE STARTED AT BYTE- 6513 PEEP#- 1 WIDTH- 16
0010 1000  0100 0101  0011 0110  0010 1001  0000 0000  0011 0110
   28H        45H        36H        29H        00H        36H
   40         69         54         41         0          54
```

Figure 6 continues

Figure 6 continued

```
  (          E          6          )                     6
 10432    17704      13893      10550      41        13824
*****************************************************************
TRANSLATE STARTED AT BYTE- 6606 PEEP#- 5 WIDTH- 16
0100 0100  0101 0000  0000 0000  0101 0100  0011 0010  1101 0101
   44H        50H        00H        54H        32H        D5H
   68         80         0          84         50         213
 D          P                    T          2          =
 17462    20548      80         21504      12884     -10958
*****************************************************************
TRANSLATE STARTED AT BYTE- 6699 PEEP#- 8 WIDTH- 16
0011 0100  0011 0000  0011 1010  0101 0111  0100 1100  0010 0100
   34H        30H        3AH        57H        4CH        24H
   52         48         58         87         76         36
 4          0          :          W          L          $
 13525    12340      14896      22330      19543      9292
*****************************************************************
TRANSLATE STARTED AT BYTE- 6793 PEEP#- 3 WIDTH- 16
1101 0101  0010 0010  0011 0001  0011 0010  0011 0011  0011 0100
   D5H        22H        31H        32H        33H        34H
   213        34         49         50         51         52
 =          "          1          2          3          4
-10972    8917       12578      12849      13106      13363
*****************************************************************
TRANSLATE STARTED AT BYTE- 6886 PEEP#- 7 WIDTH- 16
0011 0101  0011 0110  0011 0111  0011 1000  0011 1001  0011 0000
   35H        36H        37H        38H        39H        30H
   53         54         55         56         57         48
 5          6          7          8          9          0
 13620    13877      14134      14391      14648      12345
*****************************************************************
TRANSLATE STARTED AT BYTE- 6980 PEEP#- 2 WIDTH- 16
0011 0001  0011 1101  1111 1111  1111 1111  1011 1111  1111 1111
   31H        3DH        FFH        FFH        BFH        FFH
   49         61         255        255        191        255
 1          =                               USING
 12592    15665      -195       -1         -16385     -65
*****************************************************************
END OF DATA
```

line address of 17462 followed by the 16-bit binary program line number of 80 and continues with the program code of T2 = . The program code then continues in line group 5.

The operating instructions are shown on Table 1. Be sure to position the cassette tape as close as possible in front of the portion of the tape to be loaded. Only a small portion of a tape can be loaded into the buffer with each load.

For example, these tables were prepared using a default time factor of 10, which yields a clock pulse to clock pulse width of about 16 peeps. These 16 peeps represent one logical bit of information, but require two bytes of the buffer for storage. Since there are eight logical bits to a logical byte, 16 storage bytes are used for one logical byte of information. In this manner, the 7,000-byte buffer only holds about 437 logical bytes of information. By increasing the time factor during the load, a moderately higher yield can be obtained, but a time factor beyond 14 distorts the data.

The program modules are described in Table 2.

Datascope is certainly not a simple

Program Listing 1

```
10 CLEAR360:DEFINTA-Z:M1$="STARTED AT BYTE-":M$="MICROSEC"
30 PM$=STRING$(10,"*"):PM=VARPTR(PM$):PM=PEEK(PM+2)*256+PEEK(PM+
1)
35 CF=0:PE=0:BI=0:B=0:DIMDC(6):DIMDD(6):DIMDP(6)
40 V1=VARPTR(CF):V2=VARPTR(PE):V3=VARPTR(BI):V4=VARPTR(B)
50 E1=0:E2=0:E3=0:E4=0:E5=0:E6=0:F1=VARPTR(E1):F2=VARPTR(E2):F3=
VARPTR(E3):F4=VARPTR(E4):F5=VARPTR(E5):F6=VARPTR(E6)
80 T2=40:WL$=STRING$(40,"*"):BP$=STRING$(48,"*")
83 VP=VARPTR(WL$):VP=PEEK(VP+2)*256+PEEK(VP+1)-1
84 BV=VARPTR(BP$):BP=PEEK(BV+2)*256+PEEK(BV+1)
90 B2=6999:BS=25511:YS=35:NO=95:EN$="END OF DATA":AR1=32:F=5712
95 OP$="OUTPUT  1-CRT ONLY  2-CTR & PTR"
100 FA$="FACTOR":B$="H            ":Q$="HIT M-MENU C-CONT "
110 DATA128,64,32,16,8,4,2,1:DIMV5(8):FORX=1TO8:READV5(X):NEXT
135 HX$=STRING$(12,"*"):HX=VARPTR(HX$):HX=PEEK(HX+2)*256+PEEK(HX
+1)
140 DATA0,3,6,11,14,17,20,26,30,34,39,42,46,49,53,56
145 DATA58,65,70,76,79,83,87,91,96,102,108,114,120,124,128,133
150 DATA139,142,144,148,153,156,159,164,168,173,177,181,185,189,
193,199
155 DATA205,208,212,217,221,225,230,236,240,245,250,255,256,262,
264,266
160 DATA271,277,280,283,286,293,298,303,308,311,317,321,324,328,
329,330
165 DATA331,332,333,336,338,339,340,341,344,347,350,353,356,359,
362,365
170 DATA368,371,374,377,380,383,387,390,393,396,399,402,405,409,
413,417
175 DATA421,425,429,432,435,439,442,445,449,454,460,464,468
180 DIMTK(125):FORX=0TO124:READTK(X):NEXT
900 INPUT"ENTER 1-LOAD 2-ANALYZE 3-SEARCH 4-TRANSLATE";M
910 ONMGOTO1000,2000,3000,3001
920 GOTO900
1000 X=PEEK(32557):PRINT"TIME ";FA$;" =";X
1020 PRINTFA$;X;"* 6.5";M$;"+ 52.0";M$;"=";X*6.5+52.0;M$
1030 INPUT"ENTER G-GOOD OR N-NO";I$:IFI$="G"1090
1050 INPUT"NEW FACTOR 1-255";X
1070 IFX<1ORX>255THEN1050ELSEPOKE32557,X:GOTO1000
1090 INPUT"READY CASSETTE";I$
1100 PRINT"NOW LOADING":POKE16526,0:POKE16527,127:X=USR(0)
1120 PRINT"LOADING COMPLETE":GOTO900
2000 PRINT"ANALYZE ROUTINE"
2010 PRINTOP$;:INPUTP:IFP<1ORP>2GOTO2010
2020 PRINT"START ADDR 1 TO";B2;:INPUTB:IFB<1ORB>B2GOTO2020
2040 INPUT"TEST BIT WIDTH 0-NO TEST/ONE PEEP PER LINE
                          2TO 40 - TEST BIT WIDTH";T
2050 IFT<0ORT>40THEN2040ELSECLS:LA=T2
2060 PRINT"CTL KEYS   ";CHR$(92);"=AD+100  ";CHR$(91);"=AD-100
";CHR$(94);"=T+1  ";CHR$(93);"=T-1  S=STOP"
2070 PRINT"LOOKING FOR FIRST PULSE"
2080 Y=B+BS:Z=B2+BS
2085 FORX=YTOZ:IFPEEK(X)<>0GOTO2095
2090 NEXT:GOTO2900
2095 B=X-BS
2100 T8=INT(.874*T)
2110 BI=1:IFB>B2GOTO2900
2200 IF(PEEK(B+BS)ANDV5(BI))=0THENC=NOELSEC=YS
2210 IFT=0THEN2300ELSEIFC=YSGOTO2300
2230 BI=BI+1:IFBI<9THEN2200ELSEB=B+1:GOTO2110
2300 FORX=1TOLA:POKEVP+X,32:NEXT
2320 LA=1:PRINT"AD";B;TAB(8)":";CHR$(BI+48);" ";CHR$(AR1);" T";T
;
2330 IFP=2THENLPRINT"AD";B;TAB(8)":";CHR$(BI+48);" ";CHR$(AR1);"
 T";T;
2340 AR1=32
2400 POKEVP+LA,C:BI=BI+1:IFBI<9GOTO2480
2450 BI=1:B=B+1:IFB>B2GOTO2500
2480 IF(PEEK(B+BS)ANDV5(BI))=0THENC=NOELSEC=YS
2490 IFT=0GOTO2500
2492 IFLA>T8THENIFC=YSGOTO2500
2494 IFLA=T2THEN2500ELSELA=LA+1:GOTO2400
2500 PRINTTAB(18)WL$;LA:IFP=2THENLPRINTTAB(18)WL$;LA
2530 IFB>B2GOTO2900
2600 I$=INKEY$:IFI$=""THEN2300ELSEX=ASC(I$)
2635 IFX=9THEN2660ELSEIFX=8THEN2640ELSE2700
2640 T=T-1:IFT<0THEN2660ELSEAR1=93:GOTO2080
2660 T=T+1:IFT>T2THEN2640ELSEAR1=94:GOTO2080
2700 IFX=10THEN2740ELSEIFX=91THEN2720ELSE2745
2720 B=B-100:IFB<1THEN2740ELSEAR1=91:GOTO2300
2740 B=B+100:IFB>B2THEN2720ELSEAR1=92:GOTO2300
2745 PRINTQ$
2750 I$=INKEY$:IFI$=""GOTO2750
2760 IFI$="M"THEN900ELSE2300
2900 PRINT"END OF ANALYSIS":IFP=2THENLPRINT"END OF ANALYSIS"
2920 GOTO900
3000 PRINT"SEARCH ROUTINE":AR1=94:GOTO3005
3001 PRINT"TRANSLATE ROUTINE"
3005 PRINTOP$;:INPUTP:IFP<1ORP>2GOTO3005
3020 INPUT"# OF PEEPS BETWEEN CLOCK PULSES 4-40";LC
3030 IFLC<4ORLC>40GOTO3020
3040 INPUT"STARTING BYTE  1-6999";BA:IFBA<1ORBA>6999THEN3040ELSE
BA=BA-1
3053 INPUT"STARTING PEEP  1-8";BB:IFBB<1ORBB>8GOTO3053
3060 IFM=3THEN3070ELSECLS:GOTO3600
3070 INPUT"# BYTES IN SEARCH KEY 0-6 (0=NO KEY)";NB
3080 IFNB<0ORNB>6GOTO3070
3090 IFNB=0GOTO3190
3110 PRINT"ENTER DEC VAL 0-255 FOR";NB;"BYTES"
3120 FORX=1TONB
3130 PRINT"BYTE";X;:INPUTY
3140 IFY<0ORY>255THEN3130ELSEDP(X)=Y
3150 NEXT
3190 CLS:PRINT"CTL KEYS   ";CHR$(93);"=LEFT SHIFT    ";CHR$(94);
"=RIGHT SHIFT"
3200 IFNB=0GOTO3600
3210 PRINT"KEY";:FORX=1TONB:PRINTDP(X);"      ";:NEXT:PRINT
3220 IFP=2THENLPRINT"KEY";:FORX=1TONB:LPRINTDP(X);"      ";:NEXT
:LPRINT
3600 B=BA:BI=BB:GOSUB5000
```

Listing 1 continues

Listing 1 continued

```
362Ø IFM=4ORNB=ØGOTO457Ø
364Ø FORX=1TONB:IFDP(X)<>DC(X-1)GOTO457Ø
367Ø NEXT:IFP=2LPRINT"MATCH"
368Ø INPUT"MATCH FOUND   HIT C-CONT SEARCH  M-MENU";I$
369Ø IFI$="M"GOTO9ØØ
457Ø IFB>B2GOTO481Ø
458Ø I$=INKEY$:IFI$=""THEN461Ø
4582 IFM=4THEN4588ELSEX=ASC(I$)
4584 IFX=9THENAR1=94:GOTO461Ø
4586 IFX=8THENAR1=93:GOTO461Ø
4588 PRINTQ$
459Ø I$=INKEY$:IFI$=""THEN459Ø
46ØØ IFI$="M"THEN9ØØ
461Ø IFM=4THENBA=B:BB=BI:GOTO36ØØ
462Ø Y=INT(LC*.874):IFAR1=93GOTO47ØØ
463Ø FORX=1TOY:BB=BB+1:IFBB=9THENBB=1:BA=BA+1
466Ø NEXT
467Ø IFBA>B2GOTO481Ø
468Ø IF(PEEK(BA+1+BS)ANDV5(BB))=V5(BB)GOTO32ØØ
469Ø BB=BB+1:IFBB<9THEN468ØELSEBB=1:BA=BA+1:GOTO467Ø
47ØØ FORX=1TOY:BB=BB-1:IFBB=ØTHENBB=8:BA=BA-1
471Ø NEXT
472Ø IFBA<1GOTO481Ø
473Ø IF(PEEK(BA+1+BS)ANDV5(BB))=V5(BB)GOTO475Ø
474Ø BB=BB-1:IFBB>ØTHEN473ØELSEBB=8:BA=BA-1:GOTO472Ø
475Ø BB=BB-1:IFBB>ØTHEN477ØELSEBB=8:BA=BA-1:IFBA<1GOTO481Ø
477Ø IF(PEEK(BA+1+BS)ANDV5(BB))=V5(BB)THEN475Ø
478Ø BB=BB+1:IFBB<9THEN32ØØELSEBB=1:BA=BA+1:GOTO32ØØ
481Ø PRINTEN$:IFP=2LPRINTEN$
482Ø GOTO9ØØ
5ØØØ CF=INT(LC*.499):PE=INT(LC*.374)
5Ø1Ø FORX=ØTO47:POKEBP+X,48:NEXT
5Ø2Ø IFM=3THENPRINT"SEARCH ";CHR$(AR1);ELSEPRINT"TRANSLATE ";
5Ø3Ø PRINTM1$;B+1;"PEEP#-";BI;"WIDTH-";LC
5Ø4Ø IFP=1THEN5Ø5ØELSEIFM=3THENLPRINT"SEARCH ";CHR$(AR1);ELSELPR
INT"TRANSLATE ";
5Ø45 LPRINTM1$;B+1;"PEEP#-";BI;"WIDTH-";LC
5Ø5Ø POKEPM,PEEK(BV+1):POKEPM+1,PEEK(BV+2)
5Ø6Ø POKEPM+2,PEEK(V1):POKEPM+3,PEEK(V1+1)
5Ø7Ø POKEPM+4,PEEK(V2):POKEPM+5,PEEK(V2+1)
5Ø8Ø POKEPM+6,PEEK(V3):POKEPM+7,PEEK(V3+1)
5Ø9Ø POKEPM+8,PEEK(V4):POKEPM+9,PEEK(V4+1)
51ØØ POKE16526,64:POKE16527,127:X=USR(PM)
511Ø POKEV3,PEEK(PM+6):POKEV3+1,PEEK(PM+7)
512Ø POKEV4,PEEK(PM+8):POKEV4+1,PEEK(PM+9)
6ØØØ PL$=MID$(BP$,1,4)+" "+MID$(BP$,5,4)+"  "+MID$(BP$,9,4)+" "+
MID$(BP$,13,4)+"  "+MID$(BP$,17,4)+" "+MID$(BP$,21,4)+"  "+MID$(
BP$,25,4)+" "+MID$(BP$,29,4)+"  "+MID$(BP$,33,4)+" "+MID$(BP$,37
,4)+"  "+MID$(BP$,41,4)+" "+MID$(BP$,45,4)
6Ø1Ø PRINTPL$;
6Ø2Ø IFP=2LPRINTPL$
61ØØ FORX=ØTO11:Z=Ø
611Ø FORY=1TO4
612Ø IFMID$(BP$,X*4+Y,1)="1"THENZ=Z+V5(Y+4)
613Ø NEXT:IFZ<1ØTHENZ=Z+48ELSEZ=Z+55
614Ø POKEHX+X,Z
615Ø NEXT
62ØØ PL$="    "+MID$(HX$,1,2)+H$+MID$(HX$,3,2)+H$+MID$(HX$,5,2)+H
$+MID$(HX$,7,2)+H$+MID$(HX$,9,2)+H$+MID$(HX$,11,2)+"H    "
622Ø PRINTPL$;:IFP=2LPRINTPL$
63ØØ FORX=ØTO5:A=ASC(MID$(HX$,X*2+2,1)):IFA<58THENA=A-48ELSEA=A-
55
631Ø Z=ASC(MID$(HX$,X*2+1,1)):IFZ<58THENZ=Z-48ELSEZ=Z-55
632Ø DC(X)=Z*16+A:NEXT
64ØØ PRINT"  ";DC(Ø);TAB(13)DC(1);TAB(24)DC(2);TAB(35)DC(3);TAB(
46)DC(4);TAB(57)DC(5)
641Ø IFP=2THENLPRINT"  ";DC(Ø);TAB(13)DC(1);TAB(24)DC(2);TAB(35)
DC(3);TAB(46)DC(4);TAB(57)DC(5)
66ØØ FORX=ØTO5:IFDC(X)<32ORDC(X)>25ØTHENST$="        ":GOTO6729
661Ø IFDC(X)<128THENST$=CHR$(DC(X))+"      ":GOTO6729
662Ø I1$=" ":I2$=" ":I3$=" ":I4$=" ":I5$=" ":I6$=" ":I7$=" "
663Ø Y=DC(X)-128:Z=TK(Y):I1$=CHR$(PEEK(F+Z)-128)
664Ø J=TK(Y+1)-Z-1:IFJ=ØGOTO672Ø
666Ø I2$=CHR$(PEEK(F+Z+1)):J=J-1:IFJ=ØGOTO672Ø
667Ø I3$=CHR$(PEEK(F+Z+2)):J=J-1:IFJ=ØGOTO672Ø
668Ø I4$=CHR$(PEEK(F+Z+3)):J=J-1:IFJ=ØGOTO672Ø
669Ø I5$=CHR$(PEEK(F+Z+4)):J=J-1:IFJ=ØGOTO672Ø
67ØØ I6$=CHR$(PEEK(F+Z+5)):J=J-1:IFJ=ØGOTO672Ø
671Ø I7$=CHR$(PEEK(F+Z+6))
672Ø ST$=I1$+I2$+I3$+I4$+I5$+I6$+I7$
6729 ONXGOTO6731,6732,6733,6734,6735
673Ø S1$=ST$:GOTO674Ø
6731 S2$=ST$:GOTO674Ø
6732 S3$=ST$:GOTO674Ø
6733 S4$=ST$:GOTO674Ø
6734 S5$=ST$:GOTO674Ø
6735 S6$=ST$
674Ø NEXT
675Ø PL$="  "+S1$+"    "+S2$+"    "+S3$+"    "+S4$+"    "+S5$+"
  "+S6$
676Ø PRINTPL$;:IFP=2LPRINTPL$
678Ø CF=DC(Ø):IFM=3GOTO68ØØ
679Ø POKEF6+1,PEEK(V1):PRINTE6;:IFP=2LPRINTE6;
68ØØ POKEF1,PEEK(V1):CF=DC(1):POKEF1+1,PEEK(V1)
681Ø POKEF2,PEEK(V1):CF=DC(2):POKEF2+1,PEEK(V1)
682Ø POKEF3,PEEK(V1):CF=DC(3):POKEF3+1,PEEK(V1)
685Ø POKEF4,PEEK(V1):CF=DC(4):POKEF4+1,PEEK(V1)
686Ø POKEF5,PEEK(V1):CF=DC(5):POKEF5+1,PEEK(V1)
687Ø POKEF6,PEEK(V1)
688Ø PRINTTAB(8)E1;TAB(19)E2;TAB(3Ø)E3;TAB(41)E4;TAB(52)E5
689Ø IFP=2THENLPRINTTAB(8)E1;TAB(19)E2;TAB(3Ø)E3;TAB(41)E4;TAB(5
2)E5
691Ø PRINTSTRING$(64,"*");:IFP=2THENLPRINTSTRING$(64,"*")
7ØØØ RETURN
```

Program Listing 2

```
7FØØ           ØØ1ØØ        ORG     7FØØH          ;=32512
7FØØ 11581B    ØØ11Ø BEGIN  LD      DE,SIZE        ;PREP TO PRES
ET STORAGE
7FØ3 21A863    ØØ12Ø        LD      HL,BUFF
7FØ6 3EØ1      ØØ13Ø INIT   LD      A,Ø1H          ;CONSTANT OF
ØØØØ ØØØ1
7FØ8 77        ØØ14Ø        LD      (HL),A         ;STORE THE CO
NSTANT
7FØ9 23        ØØ15Ø        INC     HL             ;ADD 1 TO ADD
RESS
7FØA 1B        ØØ16Ø        DEC     DE             ;SUB 1 FROM
SPACE
7FØB 7A        ØØ17Ø        LD      A,D
7FØC B3        ØØ18Ø        OR      E
7FØD 2ØF7      ØØ19Ø        JR      NZ,INIT        ;JUMP IF MORE
 SPACE
7FØF 21A863    ØØ2ØØ        LD      HL,BUFF        ;1ST STORAGE
 ADDRESS TO HL
7F12 11581B    ØØ21Ø        LD      DE,SIZE        ;SPACE AVAI
LABLE IN DE
7F15 3EØ4      ØØ22Ø        LD      A,Ø4H          ;CASSETTE O
N
7F17 D3FF      ØØ23Ø        OUT     (ØFFH),A
7F19 DBFF      ØØ24Ø WAIT   IN      A,(ØFFH)      ;WAIT FOR 1ST P
ULSE
7F1B 17        ØØ25Ø        RLA
7F1C 3ØFB      ØØ26Ø        JR      NC,WAIT
7F1E 7E        ØØ27Ø LOOP   LD      A,(HL)         ;3.5 LOAD BY
TE
7F1F 17        ØØ28Ø        RLA                    ;2.Ø ROTATE
PEEP INTO BYTE
7F2Ø 77        ØØ29Ø        LD      (HL),A         ;3.5 STORE B
YTE
7F21 DA267F    ØØ3ØØ        JP      C,BUMP         ;5.Ø JUMP IF
 FULL BYTE
7F24 18Ø2      ØØ31Ø        JR      FLIP           ;6.Ø
7F26 23        ØØ32Ø BUMP   INC     HL             ;ADVANCE STO
RAGE ADD 3.Ø
7F27 1B        ØØ33Ø        DEC     DE             ;REDUCE AVAI
LABLE SPACE 3.Ø
7F28 3EØ4      ØØ34Ø FLIP   LD      A,Ø4H          ;3.5 CASSETT
E FLIP FLOP
7F2A D3FF      ØØ35Ø        OUT     (ØFFH),A       ;5.5
7F2C Ø6ØA      ØØ36Ø        LD      B,TIME         ;3.5 LOAD DE
LAY FACTOR
7F2E 1ØFE      ØØ37Ø        DJNZ    $              ;4.Ø    TIME
>1*6.5
7F3Ø 7A        ØØ38Ø        LD      A,D            ;2.Ø
7F31 B3        ØØ39Ø        OR      E              ;4.Ø
7F32 28Ø6      ØØ4ØØ        JR      Z,DONE  ;3.5 JUMP IF NO SPAC
E AVAILABLE
7F34 DBFF      ØØ41Ø        IN      A,(ØFFH)       ;5.5 TAKE A
PEEP
7F36 17        ØØ42Ø        RLA                    ;2.Ø ROTATE
PEEP TO CARRY
7F37 C31E7F    ØØ43Ø        JP      LOOP           ;5.Ø  LOOP T
OTAL=58.5
7F3A 3EØØ      ØØ44Ø DONE   LD      A,ØØH          ;   MICROSEC
 WHEN TIME=1
7F3C D3FF      ØØ45Ø        OUT     (ØFFH),A
7F3E C9        ØØ46Ø        RET
ØØØA           ØØ47Ø TIME   EQU     ØAH            ;DEFAULT OF
1Ø
1B58           ØØ48Ø SIZE   EQU     1B58H          ;7ØØØ BYTES
OF SPACE
63A8           ØØ49Ø BUFF   EQU     63A8H          ;START BUFFE
R AT 25512
7F3F ØØ        ØØ5ØØ CTR    DEFB    ØH             ;48 BIT COUN
TER
7F4Ø CD7FØA    ØØ51Ø        CALL    2687           ;LOAD HL WIT
H ARGUMENT
7F43 E5        ØØ52Ø        PUSH    HL
7F44 DDE1      ØØ53Ø        POP     IX             ;SET PARM AD
DR IN IX
7F46 DD66Ø1    ØØ54Ø        LD      H,(IX+1)       ;MSB OF STRI
NG ADDR
7F49 DD6EØØ    ØØ55Ø        LD      L,(IX+Ø)       ;LSB
7F4C E5        ØØ56Ø        PUSH    HL
7F4D FDE1      ØØ57Ø        POP     IY             ;ADDR OF BIT
 LINE
7F4F DD46Ø6    ØØ58Ø        LD      B,(IX+6)       ;LOAD B WITH
 PEEP #
7F52 DD66Ø9    ØØ59Ø        LD      H,(IX+9)       ;MSB OF BUFF
ER OFFSET
7F55 DD6EØ8    ØØ6ØØ        LD      L,(IX+8)       ;LSB
7F58 11A863    ØØ61Ø        LD      DE,BUFF        ;LOAD BUFFER
 START ADDRESS
7F5B 19        ØØ62Ø        ADD     HL,DE          ;DEVELOPE BY
TE ADDRESS
7F5C 3EØØ      ØØ63Ø        LD      A,ØH
7F5E 323F7F    ØØ64Ø        LD      (CTR),A ;PRESET 48 BIT COUNT
ER
7F61 CDBE7F    ØØ65Ø        CALL    X1             ;CHECK FOR C
LOCK PULSE
7F64 2ØØ5      ØØ66Ø        JR      NZ,F1          ;JUMP IF FOU
ND
7F66 CDB47F    ØØ67Ø FIND   CALL    PEEPS          ;FIND THE CL
OCK PULSE
7F69 28FB      ØØ68Ø        JR      Z,FIND         ;LOOP TILL F
OUND
7F6B DD4EØ2    ØØ69Ø F1     LD      C,(IX+2)       ;LOAD CLOCK
FADE VALUE
7F6E 79        ØØ7ØØ WAITF  LD      A,C            ;LOAD A WITH
 FADE VALUE
7F6F A7        ØØ71Ø        AND     A              ;SET Z/NZ FL
AG
7F7Ø CA7A7F    ØØ72Ø        JP      Z,DIGIT        ;JUMP IF FAD
E = ZERO
```

Listing 2 continues

program, but a complete understanding is not necessary for its useful operation. As stated in the beginning of this article, it can be used to retrieve data from an otherwise unreadable tape; to check the tape passing speed of a recorder; to learn the format of an unfamiliar tape file; and to observe what

1. Answer Memory-Size prompt with 25512.
2. Load the Basic program A.
3. Load the Assembly program Peeper.
4. Press break and type run.
5. Many different cassettes can be loaded and studied without the need to reload these programs.
6. From the main menu of "Enter 1-Load 2-Analyze 3-Search 4-Translate" any of the four operations can be selected at any time.
7. Select the load operation to read from a cassette into the buffer.
   a. Select the time interval.
   b. Carefully position the cassette about one counter position in front of the segment of tape to be studied.
   c. Press Play/Press enter.
8. Select the analyze operation to visually inspect peeps and determine logical bit alignment.
   a. Specify the output media.
   b. Specify the starting byte address.
   c. Specify the starting test bit width.
   d. While the analyze is running
      1) Use the down-arrow and up-arrow keys to inspect different areas of the buffer.
      2) Use the right-arrow and left-arrow keys to try different test bit widths.
      3) Use any other keys to stop the operation.
   e. Note the byte and peep addresses of interesting areas of data.
   f. Note the actual average bit width.
9. Select the search operation to scan interpreted data to determine logical byte alignment.
   a. Specify the output media.
   b. Specify the bit width.
   c. Specify the starting byte address.
   d. Specify the starting peep address.
   e. Specify the search key.
   f. While the search is running
      1) Use the left or right-arrow keys to change the direction of the search.
      2) Use any other key to stop the routine.
   g. Note the byte and peep addresses where byte alignment is achieved.
10. Select the translate operation to interpret the data.
    a. Specify the output media.
    b. Specify the bit width.
    c. Specify the starting byte address.
    d. Specify the starting peep address.
    e. While the translate is running use any key to stop the routine.
11. When any of the routines reach the end of the buffer it automatically stops and returns to the main menu.

Table 1. Operator Guidelines

Listing 2 continued

```
7F73 CDB47F    00730        CALL   PEEPS          ;INCREMENT T
O NEXT PEEP
7F76 0D        00740        DEC    C              ;DECREMENT F
ADE VALUE
7F77 C36E7F    00750        JP     WAITF          ;LOOP TILL F
ADE = ZERO
7F7A 1630      00760 DIGIT  LD     D,30H          ;LOAD D WITH
 ASCII "0"
7F7C DD4E04    00770        LD     C,(IX+4)       ;LOAD PERIOD
 IN C
7F7F 79        00780 DIG2   LD     A,C            ;LOAD A WITH
 C
7F80 A7        00790        AND    A              ;TO SET Z/NZ
 FLAG
7F81 CA907F    00800        JP     Z,STASH        ;JUMP WHEN P
AST PERIOD
7F84 CDB47F    00810        CALL   PEEPS          ;GET NEXT PE
EP
7F87 CA8C7F    00820        JP     Z,DECR         ;Z=ZERO PEEP
(NO PULSE)
7F8A 1631      00830        LD     D,31H          ;LD ASCII"1"
 ON PULSE
7F8C 0D        00840 DECR   DEC    C              ;DECREMENT P
ERIOD
7F8D C37F7F    00850        JP     DIG2           ;LOOP TILL P
AST PERIOD
7F90 FD7200    00860 STASH  LD     (IY+0),D       ;STASH 0 OR
1 FROM D
7F93 FD23      00870        INC    IY             ;BUMP BIT FI
ELD ADDRESS
7F95 3A3F7F    00880        LD     A,(CTR)        ;LOAD A WITH
 BIT COUNTER
7F98 3C        00890        INC    A              ;BUMP BIT CO
UNTER
7F99 323F7F    00900        LD     (CTR),A        ;PUT AWAY BI
T COUNTER
7F9C FE30      00910        CP     30H            ;COMPARE TO
BIT LIMIT 48
7F9E CAA47F    00920        JP     Z,RSTOR        ;JUMP IF LIM
IT REACHED
7FA1 C3667F    00930        JP     FIND           ;LOOP TO FIN
D NEXT CLOCK
7FA4 11A863    00940 RSTOR  LD     DE,BUFF        ;LOAD DE WIT
H BUFFER START
7FA7 B7        00950        OR     A              ;TO CLEAR CA
RRY FLAG
7FA8 ED52      00960        SBC    HL,DE          ;CALC BYTE O
FFSET
7FAA DD7006    00970        LD     (IX+6),B       ;RET NEW PEE
P #
7FAD DD7409    00980        LD     (IX+9),H       ;RET NEW MSB
 BUFFER ADD
7FB0 DD7508    00990        LD     (IX+8),L       ;RET NEW LSB

7FB3 C9        01000        RET                   ;RETURN
7FB4 04        01010 PEEPS  INC    B              ;BUMP PEEP N
UMBER
7FB5 3E09      01020        LD     A,09H          ;LOAD A WITH
 LIMIT
7FB7 B8        01030        CP     B              ;COMPARE TO
LIMIT
7FB8 C2BE7F    01040        JP     NZ,X1          ;JUMP IF NOT
 LIMIT
7FBB 0601      01050        LD     B,01H          ;RESTART WIT
H PEEP 1
7FBD 23        01060        INC    HL             ;BUMP BYTE #

7FBE 78        01070 X1     LD     A,B            ;PUT PEEP# I
N A
7FBF FE01      01080        CP     01H            ;PEEP# 1 ?
7FC1 C2C77F    01090        JP     NZ,TWO         ;JUMP IF NOT
 1
7FC4 CB7E      01100        BIT    7,(HL)         ;TEST PEEP#
1
7FC6 C9        01110        RET
7FC7 FE02      01120 TWO    CP     02H            ;PEEP# 2 ?
7FC9 C2CF7F    01130        JP     NZ,THREE       ;JUMP IF NOT
 TWO
7FCC CB76      01140        BIT    6,(HL)         ;TEST PEEP#
2
7FCE C9        01150        RET
7FCF FE03      01160 THREE  CP     03H            ;PEEP# 3 ?
7FD1 C2D77F    01170        JP     NZ,FOUR        ;JUMP IF NOT
 THREE
7FD4 CB6E      01180        BIT    5,(HL)         ;TEST PEEP#
3
7FD6 C9        01190        RET
7FD7 FE04      01200 FOUR   CP     04H            ;PEEP# 4 ?
7FD9 C2DF7F    01210        JP     NZ,FIVE        ;JUMP IF NOT
 FOUR
7FDC CB66      01220        BIT    4,(HL)         ;TEST PEEP#
4
7FDE C9        01230        RET
7FDF FE05      01240 FIVE   CP     05H            ;PEEP# 5 ?
7FE1 C2E77F    01250        JP     NZ,SIX         ;JUMP IF NOT
 5
7FE4 CB5E      01260        BIT    3,(HL)         ;TEST PEEP#
5
7FE6 C9        01270        RET
7FE7 FE06      01280 SIX    CP     06H            ;PEEP# 6 ?
7FE9 C2EF7F    01290        JP     NZ,SEVEN       ;JUMP IF NOT
 SIX
7FEC CB56      01300        BIT    2,(HL)         ;TEST PEEP#
6
7FEE C9        01310        RET
7FEF FE07      01320 SEVEN  CP     07H            ;PEEP# 7 ?
7FF1 C2F77F    01330        JP     NZ,EIGHT       ;JUMP IF NOT
 7
7FF4 CB4E      01340        BIT    1,(HL)         ;TEST PEEP#
7
7FF6 C9        01350        RET
7FF7 CB46      01360 EIGHT  BIT    0,(HL)         ;TEST PEEP#
8
7FF9 C9        01370        RET
0000           01380        END
00000 TOTAL ERRORS
```

Bugs in your software sales? GOTO The Software Guild

The Software Guild has been de-bugging the marketing program. It's designed to represent you the microcomputer software developer.

The Guild diligently guides your program through our program. This starts with a comprehensive evaluation. We'll let you know if your program is appropriate, and if its marketable. If needed, we'll even assist you in rewriting the documentation. Next comes professional packaging and an innovative merchandising system to put your product in the hands of the consumer. We'll back that up with responsive customer service. You can sit at home, collect royalties, and maybe write more software.

We've set high standards for ourselves and the software products we market. The consumer will recognize that a Guild product means quality and reliability. Call us, we've de-bugged the program for you!

The Software Guild™

Contact: Regina LaRocca
(415) 887-6899 ✓53
24213 CLAWITER ROAD • HAYWARD, CALIFORNIA 94545

©1982 The Software Guild

✓See List of Advertisers on Page 563

effect different volume settings have on the pulse and gap width, allowing you to determine the optimum setting for a particular tape.

If you find other uses, I would appreciate hearing about them. ■

Dennis Ridgway lives at 2160 Frisco Avenue, Terre Haute, IN 47805.

Basic Program	
Mainline Module	
10–180	Initialize Variables
900–920	Main Operations Menu
Load Module	
1000–1070	Select Parameters
1090–1120	Call Assembler Load Routine
Analyze Module	
2000–2050	Select Parameters
2060–2090	Look for First Byte With Pulses
2095–2230	Find First Pulse
2300–2340	Prepare Printline
2400–2494	Fill Printline Until Next Clock Pulse Is Found
2500–2760	Print Line And Test Control Keys
2900–2920	End of Analysis
Search And Translate Module	
3000–3150	Select Parameters
3190–3220	Print The Control And Key Lines
3600–3620	GOSUB 5000 "The Interpretation Subroutine"
3640–3690	Test For Match
4570–4600	Test Control Keys
4610	Prepare for Next Translation
4620–4690	Shift Right for Next Search
4700–4780	Shift Left for Next Search
4810–4820	End of Data
Interpretation Module	
5000–5120	Call Assembly Routine To Convert Peeps to Bits
6000–6020	Print Bit Line
6100–6220	Convert Bits to Hex And Print
6300–6320	Convert Hex to Decimal
6400–6410	Print Decimal Line
6600–6610	Convert Decimal to Character
6620–6720	Convert Decimal to Token Words
6729–6760	Print Character and Token Line
6780–6870	Convert Decimal to Integer Values
6880–6890	Print Integer Values
6910–7000	Print Separator Line and Return
Assembly Program	
Load Subroutine	
100–190	Initialize Buffer to Hex 01s
200–260	Wait for First Pulse
270–430	Load Buffer
440–450	Cassette Off and Return
Peep To Bit Conversion Subroutine	
500–640	Initialize Registers
650–680	Find Clock Pulse
690–750	Wait Past Clock Fade
760–850	Check If 0 or 1 Bit
860–930	Store Digit and Test For Done
940–1000	Prepare for Return And Return
1010–1060	Bump to Next Peep
1070–1380	Determine if Bit On or Off

Table 2. Program Overview

Over thirty years of down-to-earth experience as a precision parts manufacturer has enabled Star to produce the Gemini series of dot matrix printers—a stellar combination of printer quality, flexibility, and reliability. And for a list price of nearly 25% less than the best selling competitor.

The Gemini 10 has a 10" carriage and the Gemini 15 a 15½" carriage. Plus, the Gemini 15 has the added capability of a bottom paper feed. In both models, Gemini quality means a print speed of 100 cps, high-resolution bit image and block graphics, and extra fast forms feed.

Gemini's flexibility is embodied in its diverse specialized printing capabilities such as super/sub script, underlining, backspacing, double strike mode and emphasized print mode. Another extraordinary standard feature is a 2.3K buffer. An additional 4K is optional. That's twice the memory of leading, comparable printers. And Gemini is compatible with most software packages that support the leading printers.

Gemini reliability is more than just a promise. It's as concrete as a 180 day warranty (90 days for ribbon and print head), a mean time between failure rate of 5 million lines, a print head life of over 100 million characters, and a 100% duty cycle that allows the Gemini to print continuously. Plus, prompt, nationwide service is readily available.

So if you're looking for an incredibly high-quality, low-cost printer that's out of this world, look to the manufacturer with its feet on the ground—Star and the Gemini 10, Gemini 15 dot matrix printers.

star
MICRONICS • INC

MAKING A NAME FOR OURSELVES

1120 Empire Central Place, Suite 216, Dallas, TX 75247
For more information, please call Bob Hazzard, Vice President, at (214) 631-8560.

Pick a Card. . .

by Norman Efroymson

This program won't pull a rabbit out of a hat, but it will amaze and puzzle your friends.

"Pick any number up to 255." This line begins a magic trick where the subject picks a number within a given range, without revealing his choice to the magician. The magician displays several cards filled with rows of numbers and asks the subject to point out which cards contain his secret number. The magician looks at these cards for a moment, performs some hand waving and incantations, and reveals the number that the subject picked.

The solution is simple; the numbers on the cards are arranged from smallest to largest. The magician adds the numbers in the upper left corner of each card on which the subject says his number appears. That sum equals the number picked by the subject. In discussing how the trick works, see Fig. 1.

The cards are constructed in the following manner: All numbers that have bit 0 set (equal to 1) appear on card 0. All numbers that have bit 1 set appear on card 1 and so on. The bit position values are 1, 2, 4, 8, 16, . . . (from right to left) and these are the smallest numbers to appear on cards 0, 1, 2, 3, 4, . . . respectively.

If the subject chooses number 13, whose binary representation is 00001101, the 13 is displayed on cards 0, 2, and 3. The first numbers on these cards are 1, 4, and 8, which add up to 13.

For the computer to play the magician's part, it must be able to construct the series of cards. Fortunately, the logical operator, AND, allows the computer to test the bit positions of each number. For example, should 5 be printed on card 2? Card 2 holds all numbers with bit 2 set. The value of bit 2 is 2^2, or 4. We need to test bit 2 of the number 5 to see if it is set. This is determined by the Basic operation: 4 AND 5. If 5 has bit 2 set, the operation returns a true (−1). If not, it returns a false (0). If X represents the number in question and Y the value of the bit position then IF X AND Y THEN PRINT X will print X on the card.

We are now ready to prepare a program that will play magician. Two loops are needed, one to count the cards and one to print the numbers on the cards. If N cards are used, the outer loop must range from 2^0 to 2^{N-1}. The inner loop ranges from 1 to 2^{N-1}.

Lines 50 and 130 form the loop that counts the number of cards to be printed. Lines 70 and 90 count the number of numbers to be tested, and if possible, to be printed on each card. Line 80 tests the bit position and prints the number if the bit position is set. Line 120 adds the value of the bit position (which is also the first number on the screen) to the total. The drawback to this system is that it's *slow* for several cards. You could take care of this by writing the program in machine code. Program Listing 2 contains the source and object codes for Magic Card.

Lines 180–200 give introductory information. Line 260 puts a one in the B and C registers. The B register is the equivalent to the variable Y in the Basic program, and holds the value of the bit position. The C register is equivalent to X and cycles through all the numbers in the range. Lines 270–290 clear the screen before printing each card. Lines 300–320 do the equivalent of the X AND Y operation in the Basic program. Line 330 routes a false answer to line 490. If true, the number is printed on the screen, so line 340 transfers the binary number from the C register to the L register to make use of the ROM subroutine that con-

Number	Binary 7	6	5	4	3	2	1	0	
	7	6	5	4	3	2	1	0	= bit positions
1	0	0	0	0	0	0	0	1	
2	0	0	0	0	0	0	1	0	
3	0	0	0	0	0	0	1	1	
4	0	0	0	0	0	1	0	0	
5	0	0	0	0	0	1	0	1	
10	0	0	0	0	1	0	1	0	
15	0	0	0	0	1	1	1	1	
20	0	0	0	1	0	1	0	0	

Figure 1

The Key Box

Model I
16K RAM
Basic Level II

```
10  CLS
20  INPUT "PICK THE NUMBER OF CARDS TO BE USED (UP TO 8)";N
30  PRINT "PICK A NUMBER BETWEEN 0 AND";2[N-1
40  INPUT "PRESS <ENTER> WHEN READY";A$
50  FOR Y=0 TO N-1
60  CLS: PRINT "CARD #";Y
70  FOR X=1 TO 2[N-1
80  IF X AND 2[Y THEN PRINT X;
90  NEXT X
100 PRINT
110 INPUT "IS YOUR NUMBER ON THE SCREEN";A$
120 IF LEFT$(A$,1) = "Y" THEN T=T+2[Y
130 NEXT Y
140 PRINT "YOUR NUMBER WAS";T
```

Program Listing 1

"Prepare a program that will play magician."

verts binary numbers in the HL pair into their ASCII decimal equivalents. Line 350 calls the subroutine which does this.

Upon return, lines 360–450 make sure the number printed to the screen won't overrun the end of a line. Since the range of numbers used in this program is 255, the greatest number of digits a number can have is three. In addition, every number has a trailing space. So allow for four characters. If a number is set to be printed in the last four print positions of a line, it is shifted to the start of the next line. Line 360 gets the cursor position and puts it into the DE register pair for manipulation. Line 370 puts the LSB of the cursor location (a number between 0 and 63) into the A register. Lines 380 and 390 test to see if the position is less than 60. If so, the program jumps to line 460; otherwise lines 400–450 start a new line.

Lines 590–630 ask the user if his number appeared on the screen. If the number was on the screen, lines 640–660 add the value of the position to the total. Line 670 checks to see if all the cards have been printed. If so, the program jumps to the closing routine at line 720; if not, line 690 advances B to the next bit position. Line 700 sets C equal to B because the lowest number placed on a card is the value of the bit position. Line 710 loops the program to print the next card.

Lines 850–900 contain the ROM subroutine to convert a binary number to its ASCII equivalent. Line 910 contains the tally buffer. Lines 930–1050 contain the various messages to be printed.

You may alter the program to allow the user to pick the number of cards, and the number range. ■

Norman Efroymson can be reached at 2976 Chadbourne Road, Shaker Heights, OH 44120.

```
                00100 ;**********************************
                00110 ;**      MAGIC CARD PROGRAM       **
                00120 ;**     BY  NORMAN EFROYMSON      **
                00130 ;**     2976  CHADBOURNE  RD      **
                00140 ;**  SHAKER HTS.,  OHIO   44120   **
                00150 ;**          JULY 9, 1981         **
                00160 ;**********************************
7D00            00170          ORG     7D00H            ;ENTRY = 32000
7D00 CDC901     00180 START    CALL    01C9H            ;CLEAR SCREEN
7D03 219A7D     00190          LD      HL,MSG           ;GET ADDRESS OF MSG
7D06 CDA728     00200          CALL    28A7H            ;PRINT IT
7D09 CD4900     00210 WAIT     CALL    49H              ;WAIT UNTIL KEY PRESSED

7D0C FE01       00220          CP      01H              ;IS IT <BREAK>?
7D0E CA191A     00230          JP      Z,1A19H          ;IF YES, JP TO BASIC
7D11 FE0D       00240          CP      0DH              ;IS IT <ENTER>?
7D13 20F4       00250          JR      NZ,WAIT          ;IF NOT, TRY AGAIN
7D15 010101     00260          LD      BC,0101H         ;INITIALIZE LOOP COUNTE
RS
7D18 C5         00270 LOOP1    PUSH    BC               ;SAVE LOOP COUNTERS
7D19 CDC901     00280          CALL    01C9H            ;CLEAR SCREEN
7D1C C1         00290          POP     BC               ;RESTORE LOOP COUNTERS
7D1D 78         00300 LOOP2    LD      A,B              ;PUT 2[N IN 'A' REGISTE
R
7D1E A1         00310          AND     C                ;MASK OUT OTHER BITS
7D1F B8         00320          CP      B                ;WAS BIT SET?
7D20 201E       00330          JR      NZ,CONT          ;IF SET, GET NEXT NUMBE
R
7D22 69         00340          LD      L,C              ;READY FOR CONVERT
7D23 CD8E7D     00350          CALL    DECMAL           ;CONVERT HEX TO DECIMAL

7D26 ED5B2040   00360          LD      DE,(4020H)       ;GET CURSOR POSITION
7D2A 7B         00370          LD      A,E              ;GET LSB
7D2B E63F       00300          AND     3FH              ;MASK OUT HIGH BITS
7D2D FE3C       00390          CP      3CH              ;ROOM FOR LAST NUMBER?
7D2F FA3B7D     00400          JP      M,NEXT           ;JP IF ROOM
7D32 F63F       00410          OR      3FH              ;LSB = END OF LINE
7D34 B3         00420          OR      E                ;RESTORE HIGH BITS
7D35 5F         00430          LD      E,A              ;DE NOW AT END OF LINE
7D36 13         00440          INC     DE               ;DE AT START OF NEXT LI
NE
7D37 ED532040   00450          LD      (4020H),DE       ;SAVE TO CURSOR
7D3B C5         00460 NEXT     PUSH    BC               ;SAVE LOOP COUNTERS
7D3C CDA728     00470          CALL    28A7H            ;PRINT THE NUMBER
7D3F C1         00480          POP     BC               ;RESTORE LOOP COUNTERS
7D40 0C         00490 CONT     INC     C                ;INC LOOP FOR NEXT NUMB
ER
7D41 AF         00500          XOR     A                ;ZERO 'A' REGISTER
7D42 B9         00510          CP      C                ;IS 'C' REG = 0?
7D43 20D8       00520          JR      NZ,LOOP2         ;IF NOT, JP BACK TO LOO
P2
7D45 21C03F     00530          LD      HL,3FC0H         ;HL => TO LAST LINE
7D48 222040     00540          LD      (4020H),HL       ;CURSOR ON LAST LINE
7D4B 21037E     00550          LD      HL,MSG2          ;GET MESSAGE
7D4E C5         00560          PUSH    BC               ;SAVE LOOP COUNTERS
7D4F CDA728     00570          CALL    28A7H            ;PRINT IT
7D52 C1         00580          POP     BC               ;RESTORE LOOP COUNTERS
7D53 CD4900     00590 YESRNO   CALL    49H              ;WAIT FOR KEY INPUT
7D56 FE4E       00600          CP      'N'              ;IS IT 'N'?
7D58 280B       00610          JR      Z,NXTCRD         ;IF NOT, JP TO NEXT CAR
D

7D5A FE59       00620          CP      'Y'              ;WAS KEY 'Y'?
7D5C 20F5       00630          JR      NZ,YESRNO        ;IF NOT, TRY AGAIN
7D5E 3A997D     00640          LD      A,(TOTAL)        ;GET SUBTOTAL
7D61 80         00650          ADD     A,B              ;ADD BIT POSITION'S VAL
UE
7D62 32997D     00660          LD      (TOTAL),A        ;SAVE TO TOTAL
7D65 CB78       00670 NXTCRD   BIT     7,B              ;ALL CARDS BEEN PRINTER
?
7D67 2005       00680          JR      NZ,FINAL         ;GO IF YES
7D69 CB20       00690          SLA     B                ;SHIFT FOR NEXT CARD
7D6B 48         00700          LD      C,B              ;BEGIN COUNTER AT B
7D6C 18AA       00710          JR      LOOP1            ;BACK TO PRINT CARD
7D6E CDC901     00720 FINAL    CALL    01C9H            ;CLEAR SCREEN
7D71 21267E     00730          LD      HL,MSG3          ;GET MESSAGE
7D74 CDA728     00740          CALL    28A7H            ;PRINT IT
7D77 3A997D     00750          LD      A,(TOTAL)        ;GET THE ANSWER
7D7A 6F         00760          LD      L,A              ;READY FOR CONVERT
7D7B CD8E7D     00770          CALL    DECMAL           ;CONVERT IT TO DECIMAL
7D7E CDA728     00780          CALL    28A7H            ;PRINT THE ANSWER
7D81 AF         00790          XOR     A                ;ZERO 'A' REGISTER
7D82 32997D     00800          LD      (TOTAL),A        ;ZERO TOTAL BUFFER FOR
                00810                                   ;NEXT GAME
7D85 21B27D     00820          LD      HL,MSG1          ;GET MESSAGE
7D88 CDA728     00830          CALL    28A7H            ;PRINT IT
7D8B C3097D     00840          JP      WAIT             ;RESTART GAME
7D8E C5         00850 DECMAL   PUSH    BC               ;CONVERT ROUTINE
7D8F 2600       00860          LD      H,0              ;HL HAS HEX NUMBER
7D91 CD9A0A     00870          CALL    0A9AH            ;SAVE AS INTEGER
7D94 CDBD0F     00880          CALL    0FBDH            ;CONVERT IT TO DECIMAL
7D97 C1         00890          POP     BC               ;RESTORE LOOP COUNTERS
7D98 C9         00900          RET
7D99 00         00910 TOTAL    DEFB    0
7D9A 17         00920 MSG      DEFB    23               ;32 CHAR/LINE MODE
7D9B 20         00930          DEFM    '  M A G I C   G U E S S'
7DB2 0A0A       00940 MSG1     DEFW    0A0AH            ;2 LINE FEEDS
7DB4 54         00950          DEFM    'THINK OF A NUMBER (0-255)'
7DCD 0A         00960          DEFB    10
7DCE 50         00970          DEFM    'PRESS <ENTER> WHEN READY.'
7DE7 0A         00980          DEFB    10               ;FOR BASIC
7DE8 4F         00990          DEFM    'OR PRESS <BREAK> FOR BASIC'
7E02 00         01000          DEFB    0
7E03 49         01010 MSG2     DEFM    'IS YOUR NUMBER IN THIS LIST (Y/N)?'
7E25 00         01020          DEFB    0
7E26 17         01030 MSG3     DEFB    23
7E27 54         01040          DEFM    'THE NUMBER YOU PICKED WAS'
7E40 00         01050          DEFB    0
7D00            01060          END     START
00000 TOTAL ERRORS
```

Program Listing 2. Source and Object Codes.

AND. . .OR. . .NOT

by Jeffrey Myers

Do you feel comfortable using Boolean logic? Or do the words AND, OR, and NOT scare you? This article explains just how they are used.

Just beyond beginner's Basic at the heart of your TRS-80's architecture and its machine language lurk creatures with which you should become familiar. They populate a branch of the mathematics family tree which belongs to the logic family. These basic beasts are named AND, OR and NOT.

Fortunately, AND, OR and NOT are about as fearsome as they look—harmless words that we English speakers have long since mastered. In logic (as in mathematics or computer science) they have precise definitions which distinguish them from the everyday AND, OR and NOT.

Perhaps you have seen them in your Basic manual or in someone else's programs. One such usage is very straightforward.

```
100 IF A = 5 AND B = -6 THEN 111
```

This Basic statement transfers program control to line 111 if A is 5 and B is −6. If either equation is false or if both are false, the program flows to the statement below line 100. Here, AND functions as you might guess. But what about the statement

```
200 IF A AND (B AND C) THEN 222
```

in which there are no relation symbols such as "=" or "<"? The answer will have to wait a bit, but it will reveal a fascinating world of programming techniques.

OR appears much as the ANDs above and AND and OR can appear together in statements with predictable consequences such as:

```
300 IF (A = 1 AND B = 10) OR (A = 10 AND
B = 1) THEN 333
```

This statement can send you to line number 333 in two ways: A and B equal 1 and 10 respectively, or vice versa. In mastering these logical operators one of the challenges is determining which if any parentheses to eliminate in line 300. This is beginning to sound like algebra, and for a good reason.

Anywhere they occur, AND and OR are binary operations. (This is not base two numeration.) It means that they must operate on two things at a time and the things are called operands. In ordinary arithmetic, addition, subtraction, multiplication and division are binary operations. Procedures exist for adding columns of numbers at once, but fundamentally these techniques still add two numbers at a time. (Adding columns in various directions and combinations is reliable because addition has two special properties called commutativity and associativity. Subtraction has neither of these properties.)

By contrast, unary operations such as taking the square root of a number require only one operand. The term unary also refers to our third logical operator, NOT, which is performed only on single operands. The results can look mysterious at first.

Experiment with NOTing numbers by entering commands like PRINT NOT 2 or PRINT NOT 10000 and so forth. Try enough numbers to allow a pattern to emerge and note what numbers are out of bounds for the operation NOT.

For legal values, a simple pattern emerges quickly, although the reason for calling this operation NOT is probably still obscure. NOT 3 is −4. NOT −4 brings you back to 3. Although this alone hardly proves the fact, this does work in general: If NOT A is B then NOT B is always A. Mathematically, NOT is said to be its own inverse.

NOT treats decimal inputs as if the Basic INT function had been performed first. Instead of operating on the decimal value, NOT uses the largest integer less than or equal to it. This is a clue to the nature of NOT.

The lowest positive integer which produces an error message is 32678. The integer 32767 works and produces a −32768. The lowest acceptable integer is −32768 and when NOTed, it produces 32767. These are the exact limits placed on integer variables in Basic. You declare integer variables by

```
10 INPUT X
20 IF X THEN 50
30 PRINT"DROPPED THROUGH"
40 GOTO10
50 PRINT"BRANCHED"
60 GOTO10
```

Program Listing 1

The Key Box

Basic Level II
Model I or III
4K RAM

DEFINT or "%".

In Basic, integer variables can be stored internally in two bytes (16 binary digits) and most commonly, the highest digit shows the sign of the number with zero and one representing positive and negative respectively. The number 32767 is simply binary 0111111111111111, the highest number that fits this numeration scheme. In this context, binary means base two.

Two's Complement

Negatives are tricky since Basic stores them in a special form called *two's complement*. Briefly, the *one's complement* of a binary number has all bits (binary digits) opposite to those of the original number. This form is inefficient for operating with negatives, however. More popular is *two's complement* which is the *one's complement* plus one.

In binary form, the number one (0000000000000001) has a one's complement of 1111111111111110. Computers could use this internally to store negative one, but if the binary forms of 1 and −1 are added, 1111111 111111111 results. This would equal −32767 under this plan. To use one's complement and the common addition algorithm, we would have to agree that 1 + (−1) = −32767!

Using two's complement, negative one is one added to the one's complement, giving 1111111111111111. This, when added to one (0000000000 000001) gives 0000000000000000 with a carry of 1. Ignore the carry and the rest of the answer is always correct! This type of operation is at the heart of integer arithmetic and is the reason computers can perform at high speed.

In light of the preceding information, NOT is one's complementer. It alters every bit of the binary form from 0 to 1 or 1 to 0.

AND and OR also perform bit-by-bit operations on the binary representations of numbers. This will open the door to conditional statements such as

```
400 IF P AND Q THEN 444
```

To understand this powerful programming tool, the If . . . Then statement needs some attention.

When interpreting Basic statements such as IF J = 34 THEN 55, you make a judgment about the mathematical sentence "J = 34". (Mathematical sentences include words like =, <, and so on.) In short, the computer assigns a truth value to it, using 0 for false and −1 for true. The flow of the program hinges on whether the hypothesis has a value of 0. If the truth value is zero, the program continues in numerical order. Otherwise the branch specified is performed. Verify this with a test program such as Program Listing 1. Only one input will cause the computer to print "DROPPED THROUGH". The key number for which the computer actually tests is zero.

The −1 commonly used for true is NOT 0 and has all ones in its two's complement signed binary form. As illustrated in Listing 1, other non-zero values function the same within a conditional statement.

In symbolic logic, NOT, AND, and OR are defined by the truth tables such as those shown in Table 1. Here, assume P and Q are statements whose truth can be determined and assume 0 = false and 1 = true. (Letting 1 stand for true is more common in logic and in some non-TRS-80 computers.)

By definition P AND Q is true only when *both* statements are true. P OR Q is true if P is true, if Q is true or if both are true. NOT reverses the truth value.

Consider the following Basic statement:

```
500 LET LO = −A*(A<B)−B*(A>=B)
```

This statement does not produce an error message when used in a program! In ordinary algebra, this might be meaningless, but to the Basic interpreter, it makes sense. On top of that, it performs a useful function: LO will be set equal to either A or B, whichever is numerically less.

If A is 1 and B is 2, the sentence in the first pair of parentheses is true and the other sentence is false. The computer will, in this type of statement, compute a truth value and substitute it for each expression. In this example, it will assign LO a value equal to −A*(−1)−B*(0) which equals A. If B is the smaller number, the first expression is false and the second is true. If A and B are equal, LO is assigned B's value (although this is arbitrary).

Altering line 500 to print the maximum of two numbers requires only slight changes. What about the maximum of three numbers? Try this as a programming challenge. A clue will emerge later.

P	Q	NOT P	NOT Q	P AND Q	P OR Q
1	1	0	0	1	1
1	0	0	1	0	1
0	1	1	0	0	1
0	0	1	1	0	0

Table 1. Truth table definitions

Basic Mathematics

Basic's method of dealing with mathematical sentences explains why an incorrectly typed statement like

```
600 LET X=Y=Z
(where you meant LET X=Y−Z)
```

produces no error message, but assigns 0 or −1 to X depending upon whether Y and Z are unequal or equal. By the nature of a Let statement, the X must be the storage place for the computed variable. The computer then tries to compute Y = Z. Because this is a sentence rather than just an expression, it assigns a truth value to it. When the algebra of real numbers fails, TRS-80 Basic just shifts gears into the algebra of logic.

The Basic statements IF X<>0 THEN 99 and IF X THEN 99 perform the same task. The main difference is in execution speed. The speed advantage is important inside loops that are executed many times in a program.

Earlier, I gave a Basic statement that appears as line 20 in Program Listing 2. Read the program and predict the results before you read on.

This program ANDs according to the truth table definition as if the ones and zeros of the binary number were trues and falses. If you entered 1, 2 and 3 when running Listing 2, first the AND in parentheses is performed as shown in Fig. 1. This is then ANDed with one resulting in zero. It is interpreted as false and program control passes to line 30. To force the program to line 40, all three numbers must have ones in the same bit somewhere to make the answer non-zero. Three equal numbers do this. Inputs 2, 3, and 6 also make the conditional true because they are 10, 11 and 110 in binary and ANDing them results in 010. Like addition, you can AND by columns: If a column contains all ones write one, otherwise write zero.

Once again, remember that all non-zero truth values are treated as true.

During the processing of an If . . . Then statement, the real question is "Is the truth value of the hypothesis zero or not?" If any bit of the truth value is one, it is treated as true.

The binary nature of ANDing helps explain such curiosities as PRINT 6 AND 5 producing the seemingly nonsensical answer of 4. PRINT 6 OR 5 produces a 7 on the screen, the result of ORing binary 110 with binary 101. Try to predict the result of PRINT 4 AND 5 OR 6, then test it. Your Basic manual will show that AND has priority over OR just as multiplication is performed before addition. This allows simpler expressions by eliminating some parentheses. Mastering the NOT-AND-OR hierarchy is an important step in symbolic logic.

decimal	binary
2	0010
3	0011
2 AND 3	0010
1	0001
1 AND (2 AND 3)	0000

Fig. 1. ANDing in binary

AND, OR and NOT reveal some of what goes on at the machine level. Basic and other higher-level languages go to great lengths to accommodate base-10 numeration but meanwhile, the Z80 microprocessor hums along in base two. When a friend says "Hey, I thought computers did everything in base two but it looks decimal to me!" type ?1AND2 and explain why 1 AND 2 equals zero.

```
    XXXX1XXX
AND 00001000
    00001000
```

Figure 2

```
10 INPUT A,B,C
20 IF A AND ( B OR C ) THEN 40
30 PRINT"DROPPED THROUGH":GOTO10
40 PRINT"BRANCHED":GOTO10
```

Program Listing 2

Logical Operators

One of the powerful uses of logical operators is exemplified by the statement

```
700 IF PEEK(14400) AND 8 THEN 777
```

Each byte of memory consists of eight bits. If the highest bit is not a sign bit, decimal numbers from 0 to 255 can be stored in one byte. Commonly, the eight bits are numbered 0–7 from least to most significant.

The statement above helps spot whether the up-arrow key is pressed. This key sets bit 3 (to become 1) in the contents of memory location 14400. The effect of the AND is shown in Fig. 2. At the top are the eight bits of memory location 14400 with bit 3 set. Below is the number 8 in binary. The Xs may be set or reset (0) by other keys at the moment, but none of them will matter because each will be ANDed with the zero below it. In the example shown, the result of ANDing is 00001000 which (since it is not zero) will be inter-

preted as true. If the up-arrow key is not pressed, the upper 1 will be zero and the AND operation produces 00000000 or false.

The statement IF PEEK(14400) = 8 THEN 777 is not equivalent to line 700. Any of the eight keys that affect memory location 14400 will make the hypothesis false whether or not the up arrow is pressed. In situations like this, AND is a great time-saver in Basic programming.

The technique above is called masking. The effect of AND 8 was to hide (mask) all bits except bit 3. You can mask with AND to tell if an integer is even or odd (IF X AND 1 . . .) or to find the remainder after division by 16 (IF X AND 15 . . .). Masking is especially useful in lower-level languages that are more concerned with binary numbers than decimal. This will be investigated later.

If you consider AND, OR and NOT as the building blocks of the algebra of logic, you can derive other operations from them. Table 2 is a truth table definition of the exclusive-or operation for which the symbol XOR is used.

As its name implies, P XOR Q is true only when one or the other statement is true, but not both. By contrast, regular OR can be called *inclusive-or* since it includes the possibility that both statements are true.

Basic has no symbol for XOR, but you can define it in this way:

```
800 X = (A AND NOT B) OR (B AND NOT A)
```

This could be a short subroutine, or if your Basic includes user-defined functions, you could make the above the definition of FNX(A,B). X in line 800 will be A XOR B. If you are not concerned about readability,

```
800 X=AANDNOTBORBANDNOTA
```

is equivalent since the operations order matches the existing parentheses.

Another way to accomplish *exclusive-or* is

```
800 X = (A OR B) AND NOT (A AND B)
```

You can prove that this always matches the output of the earlier line 800. In this formula of logic, dropping the parentheses radically alters its function. NOT A would then be the first operation performed, then the ANDs (left to right) and lastly the OR.

Exclusive-or has some interesting and useful properties. One technique for encoding a message involves this operation in a very straightforward way. Program Listing 3 shows this method at work.

In this program, you will code the message M$ and you must supply a coding word C$. Once the message is coded, this coding word alone will decode the message. The coding produces N$. The program will print N$ on the screen, but the coding may produce non-alphanumeric characters and the printout is unpredictable, so the ASCII codes of N$'s characters are also shown.

The scheme is as follows: First, pointers CP and MP (for C$ and M$) are set to one. Second, the ASCII codes of the CPth letter of C$ and the MPth letter of M$ are XORed and N$ gets an MPth character which has this result as its ASCII code. Third, CP and MP are incremented by one. If MP gets beyond the end of M$, the task is completed. If not, then CP is checked to see if it is beyond C$'s end. If it is, CP is set to one. Fourth, go to step two.

This leaves a string variable N$ equal in length to M$. How is the decoding done?

Decoding

Decoding is the same process with the same coding word. Once the program has shown N$, it waits for a dummy input. Press enter and Listing 3 then decodes N$ by setting M$ = N$

P	Q	P XOR Q
1	1	0
1	0	1
0	1	1
0	0	0

Table 2. Definition of XOR

```
  5 CLEAR 100
 10 INPUT"CODING WORD";C$
 20 INPUT"MESSAGE TO BE CODED";M$
 30 CP=0:MP=0:N$=""
 40 MP=MP+1:IF MP > LEN(M$) THEN 100
 50 CP=CP+1:IF CP > LEN (C$) THEN CP=1
 60 A=ASC(MID$(M$,MP,1))
 70 B=ASC(MID$(C$,CP,1))
 80 X=( A AND NOT B ) OR ( B AND NOT A )
 90 L=L+1:N$=N$+CHR$(X):GOTO40
100 PRINT N$
110 PRINT:FORI=1 TO LEN(N$):PRINT ASC(MID$,N$,I));
120 NEXT:PRINT
130 INPUT Q$:M$=N$:GOTO30
```

Program Listing 3

```
10 INPUT P,Q,R
20 A=P:B=Q:GOSUB80
30 A=X:B=R:GOSUB80:PRINT X
40 A=Q:B=R:GOSUB80
50 A=X:B=P:GOSUB80:PRINT X
60 GOTO10
80 X=(A AND NOT B) OR (B AND NOT A)
```

Program Listing 4

and jumping back to the beginning. The new N$ turns out to be the original message! This trick works because of a curious property of the *exclusive-or* operation.

You can prove in symbolic logic that if C = A XOR B then A = C XOR B. In other words, XORing with B takes you back and forth from A to C. The coding program applies this letter-by-letter. It produces a code that is not trivial to break, even if it is known that XOR is the method.

Related to this coding technique is a method for swapping the contents of two variables *without involving a third.* This is mostly a curiosity to the Basic programmer, but of more importance to Assembly-language programmers who have only a few easily accessible registers.

Here's the swapping procedure: Exclusive-or A with B and store in A; Exclusive-or this new A with B and store in B; Exclusive-or the current A and B and store in A. As if by magic, A and B have exchanged their original contents. Work through some examples by hand (with binary numbers) and watch it happen.

Exclusive-or is like AND and OR in that the order of its operands does not matter; this property is called commutativity. Another property of interest in algebra is associativity. If an operation "o" is associative, then (AoB)oC = Ao(BoC). In other words, if an operation is associative, parentheses are not necessary in a series of these operations. Both AND and OR are associative, so 15 OR 71 OR 85 OR 123 gives consistent results no matter which numbers you OR first. What does 23 AND 147 AND 92 AND 0 AND 44 equal, and how can the question be answered quickly?

Is XOR an associative operation? Will 1 XOR 2 XOR 3 depend on which pair is done first? If it is associative, could you perform a "column *exclusive-or*?" You can use Program Listing 4 or one like it to investigate. The more you try it, the more associative XOR seems to be. How long should you run the program to guarantee that it is associative?

Proving that (A XOR B) XOR C = A XOR (B XOR C) is mathematically valid falls into the realm of algebraic logic. The foundation laid down so far qualifies symbolic logic to be called a Boolean Algebra or a Boolean Ring. Within this framework, you can prove certain simplification techniques and use XOR, AND, OR, and NOT to fuller advantage.

A Boolean Ring is an abstract mathematical entity within which many theorems have been proven. These laws are then applied in many ways: to symbolic logic (of interest here), to set theory (with intersection instead of AND, complements instead of NOT, and so on), to switching theory (where one and zero are closed and open switches, AND is two switches in series, and so on), and to many other applications. The fundamental rules for all these areas are precisely the same.

A AND 0 = 0 A OR 1 = 1 A AND 1 = A A OR 0 = A	LAWS OF OPERATIONS WITH 0 AND 1
A AND A = A A OR A = A	IDEMPOTENT LAWS
NOT (NOT A) = A	LAW OF DOUBLE NEGATION
A OR NOT A = 1 A AND NOT A = 0	LAWS OF COMPLEMENTARITY
A OR B = B OR A A AND B = B AND A	COMMUTATIVE LAWS
(A OR B) OR C = A OR (B OR C) (A AND B) AND C = A AND (B AND C)	ASSOCIATIVE LAWS
A OR (B AND C) = (A OR B) AND (A OR C) A AND (B OR C) = (A AND B) OR (A AND C)	DISTRIBUTIVE LAWS
NOT (A OR B) = NOT A AND NOT B NOT (A AND B) = NOT A OR NOT B	DeMORGAN'S LAWS

In any valid logical equation, if all ones and zeros are exchanged and all ANDs and ORs are exchanged, the resulting sentence will also be valid.

Table 3. Fundamental Principles of Boolean Algebra

A	B	A OR B	NOT A	NOT B	NOT (A OR B)	(NOT A) AND (NOT B)
1	1	1	0	0	0	0
1	0	1	0	1	0	0
0	1	1	1	0	0	0
0	0	0	1	1	1	1

Table 4

Basic Laws

Table 3 lists some of the basic laws of symbolic logic. You can prove these using truth tables since the variables can only stand for zero or one. With these same rules, circuit designers have laid out and simplified the Z80 chip. Here is a facet appearing in computer hardware and software alike—Boolean Algebra!

Especially notice the powerful Principle of Duality and how it relates to the pairing of entries in this table. Each logical statement has a dual in which all ANDs are changed to ORs, all ORs to ANDs, all ones to zeros and all zeros to ones. One of the classic techniques of simplifying electronic circuits is to form the dual of its Boolean expression, simplify it using the various laws, then form its dual, which will be equivalent to the original circuit.

You can prove theorems such as NOT(A OR B) = (NOT A) AND (NOT B) by a truth table (see Table 4) which considers all combinations of truth values for A and B. Moving to the right, you fill in the columns on the basis of earlier columns and in accordance with the definitions of the operations. The fact that the final two columns are equivalent proves this theorem.

The theorem above relates to everyday logic in a simple way. The left side of the equation indicates that we are negating (making negative) an OR statement, like "It is not true that I will

either go hiking or swimming." You can simplify this cumbersome statement by forming an AND composed of negations of the inner statements: "I will not go hiking and I will not go swimming." To differing degrees the various theorems can apply to logical reasoning patterns.

In truth-table proofs, eight rows are necessary if three variables are involved; 16 are required for four variables, and so on. This makes the truth-table method tedious for large numbers of variables. Algebraic techniques therefore have their place in simplifying logical expressions (or electronic circuits).

Earlier, I gave two different logical expressions that performed the exclusive-or operation on A and B. Table 5 shows a truth-table proof of this. Again, the last two columns are the crux of this proof. The codes 1–9 make the table hard to read, but printing full headings over the columns wastes space.

An alternative to the truth table is to apply various theorems of Boolean Algebra to transform one expression into the other. Table 6 shows one such sequence of steps and justifications which proves the equivalence of (A AND NOT B) OR (B AND NOT A) with (A OR B) AND NOT (A AND B). In the proof, the terms in parentheses indicate how that statement was derived from the statement preceding it.

Every step in the algebraic proof is logically equivalent to every other step, so in writing a Basic program involving exclusive-or, you may choose the one that executes fastest or is easiest to type or easiest to comprehend, depending on your priorities. In electronics where each AND, OR and NOT is accomplished by transistors, diodes, and so on, the decision would probably fall between speed of execution and cost.

At the heart of TRS-80 Models I, II and III lies a Z80 microprocessor. In Z80 Assembly language, the techniques mentioned in this article become all the more useful because of the data's binary nature. Below are some simple illustrations.

- The commands AND, OR, XOR exist.
- The command CPL does the job of NOT or one's complementing and NEG does two's complementing.
- Most computations are performed in the accumulator, a one-byte register. The carry flag is one bit which is set to one if the last operation generated a carry (or a borrow). The commands AND A and OR A are famous tricks for resetting the carry flag to zero without changing A's contents (each command has A operating on itself). XOR A is a quick way to zero A.
- The blinking asterisk during cassette loads on Models I and III is accomplished by getting the contents of memory location 15423, XORing with 10, storing the result back in 15423, then repeating. (This is another use for XOR: to toggle or flip-flop bits.
- AND, OR, and NOT and related functions are mainstays of the machine-language routines that make Basic work. In general, a thorough knowledge of logical algebra is a prerequisite for succeeding in Assembly-language programming.

Perhaps this article has inspired you to do some further investigations into the algebra of logic. If so, your computer can teach you a lot. Also, books about Boolean Algebra, switching circuits, logic, or set theory might help. The game WFF 'N PROOF is based on these concepts. (A wff is a well-formed formula composed of the type of ingredients mentioned above.)

Here are some suggested areas of exploration:

- In graphics, PEEKing at one of the video memory locations, then ANDing or ORing with various powers of two and POKEing it back will set and reset individual pixels.
- The NAND function is an AND followed by a NOT. It has been proven that any logical formula can be rewritten using only NANDS!
- Ditto for NOR.
- The listing of laws of symbolic logic only scratches the surface. Develop further formulas. Example: How can you safely remove the parentheses from A XOR (B AND C)?
- If you need to define a function which for an input of X gives SQR(X) if X is above four but which gives 5*X−18 for other Xs, try Y = −(X>4)*(SQR(X)) − (X<=4)*(5*X−18). This type of technique executes more quickly than a lot of If...Thens. You can transplant this technique into many programming situations.
- Here's a partial answer to the challenge question of having the computer determine the highest of three numbers A,B, and C:

```
M = −A*(A>B AND A>C) −B*(B>A AND B>C) −C*(C>A AND C>B).
```

This works unless all three inputs match. Fix the formula without using If.

- As a final challenge, use logical algebra to prove that (A XOR B) XOR B = A by using the formulas given for XOR and simplifying the left side until it equals A. ■

A	B	1	2	3	4	5	6	7	8	9
1	1	0	0	1	1	0	0	0	0	0
1	0	0	1	1	0	1	0	1	1	1
0	1	1	0	1	0	0	1	1	1	1
0	0	1	1	0	0	0	0	1	0	0

1 = NOT A
2 = NOT B
3 = A OR B
4 = A AND B
5 = A AND NOT B
6 = B AND NOT A
7 = NOT (A AND B)
8 = (A AND NOT B) OR (B AND NOT A)
9 = (A OR B) AND NOT (A AND B)

Table 5. Truth table proof of two XOR formulas

1. (A & −B) OR (B & −A)
2. ((A & −B)OR B)&((A & −B)OR −A)
   (distributive law of OR over AND)
3. ((A OR B)&(−B OR B))&((A OR −A)&(−A OR −B))
   (distr. law of OR over AND, twice)
4. ((A OR B)& 1) & (1 & (−A OR −B))
   (law of complementarity)
5. (A OR B) & (−A OR −B)
   (law of ANDing with 1)
6. (A OR B) & NOT(A AND B)
   (DeMorgan's Law)

Table 6

Jeffrey Myers (210 Park Ave., Hillsboro, OH 45133) has taught high school mathematics for 12 years.

✓See List of Advertisers on Page 563

Permanent Sound

by Richard C. McGarvey

Until Richard put a permanent audio amplifier into his CRT case, his desk was a tangle of wires. Now you can be more organized, too.

With two TRS-80s on one desk and all the associated clutter, I was getting tired of having the jumble of wires needed to hook up my amplifier for the sound effects that come with many of my games and other programs. It was only after opening my CRT that I realized there is more than enough room inside the computer's casing for a permanent audio amplifier.

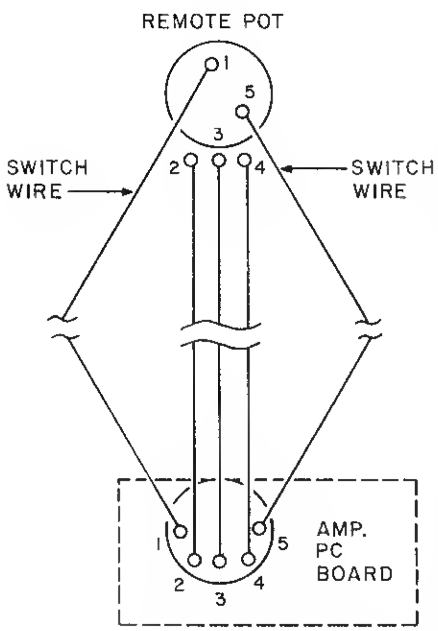


Fig. 1. Wire Set-up Between Pot and PC Board

The modification should take only 30 minutes or so. Even the inexperienced hobbyist can easily make the installation. Read the instructions before starting and be sure you have all the parts and tools ready before you begin.

On Being Careful

The CRT casing is made of soft plastic. Drilling into plastic can be troublesome, so make a pilot hole using a very fine drill bit. Increase bit sizes in small increments until you get the correct hole size. It would be better to use a hand drill or a drill bit held in a pair of vise-grip pliers rather than a power drill.

Be sure you have correctly located all holes to be drilled *before drilling*. Check and double check before you drill. You cannot undrill a hole.

You will be working inside a tv set. High-voltage residue can cause serious electric shock. Picture tubes are fragile and can cause serious injury if they implode. Be careful!

Assembly

Assemble the potentiometer and potentiometer switch as indicated on the package backs. These components will be your remote on/off and volume controls. Color coding is important at this stage. You will need at least four colors of light-gauge wire.

On the rear of the completed potentiometer are two connection points (Fig. 1). These are the on/off switch connections. Cut two lengths of red wire long enough to reach from the pot to the amplifier board; 10 inches should be long enough. Connect one wire to each of the rear connect points and solder.

Next, cut three wires of different colors the same length as the red switch wires. Attach one of these wires to each contact on the pot and solder. Twist or braid these wires into a cable for neatness. Leave an inch or so unbraided on each wire; strip the insulation from their ends and set it aside.

Now open the amplifier case. The PC board is small and can be removed by removing one screw, the volume knob, and the jack nuts on the outside of the case. Save the parts for later use. Remove the speaker also. It is fastened with a screw and gum-type glue. Remove the screw and carefully pry away the speaker.

For the next step, the power hook up, you have two options. To use a 9-volt battery, extend the power wires through a hole in the back of the CRT and mount a battery clip to hold the battery. This will make battery changing simple.

The Key Box
Model I

I use a power supply. The amplifier is designed to handle 9 volts dc. I can handle up to 15 volts without burning or as little as 5 volts dc. In either case, you will have to extend the wires.

Extending the Wires

You will have to extend the wires at two locations: between the PC board and power connect point and between the PC board and speaker. It is better to replace the wires rather than to splice on extra.

To do this, cut wires long enough to reach from the intended PC board location to the speaker and power connect. Next, unsolder one end of a wire to be extended and solder the longer wire in its place. Then unsolder the other end of the short wire and solder the other end of the replacement wire in its place. Do this until all four wires have been extended. Again, twist the wires for neatness. If you remove all the wires at once, you will most likely forget where they belong.

Installation

The easiest location for the pot on/off switch is on the same side of the CRT casing as the amplifier. This requires drilling one hole.

To install the pot with the knob on the front of the CRT (Photo 1), open the CRT back carefully. Don't worry if the inside of your TRS-80 looks different than mine (Photo 2); there are different versions.

First, drill the pilot hole for the pot shaft from the outside. Drill exactly between the S and the - in TRS-80. This will neatly center the control knob. Continue to enlarge the hole until the pot shaft fits through.

Now cut a rectangular piece of plastic from the amplifier casing that will span the hole on the inside of the CRT. This piece will be used to mount the pot (Fig. 2). Drill a hole in the piece large enough to fit the pot shaft and thread. Put the washer and nut on the pot and mount it securely to the plastic piece.

Insert the pot shaft through the CRT casing hole so the plastic mounting piece is flush with the inside of the casing. Using a small drill bit, drill a hole through one side of the mounting piece and into the casing. Be sure the drilled hole is slightly smaller than the screws you removed from the PC board and speaker. You will use these screws to mount the pot to the casing.

Screw the mounting piece and pot in place. Make sure the pot shaft is centered, and then drill the hole through the other side of the mounting bracket. Do not install the screw yet. Extend the pot far enough away from the front of the CRT so the knob does not scrape when turned. Mark this spot on the pot shaft. Remove the pot and cut the shaft at the mark. Be sure to clamp the pot by the shaft for cutting and not by the pot itself.

Decide where you want to connect the amplifier PC board, and drill a small pilot hole for one of the jacks. On the outside of the CRT, place the PC board on the side of the CRT

Photo 1. Completed installation. Notice the centered control knob directly below the Radio Shack logo; the black speaker grating for covering speaker holes; and the marked audio and power jacks on the casing's right side.

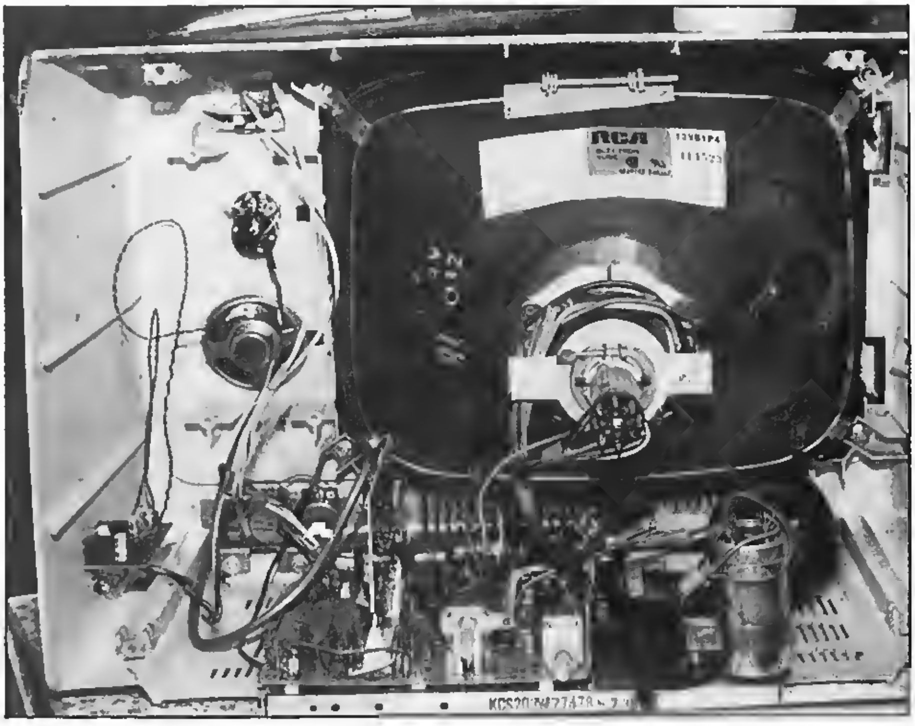


Photo 2. Opened casing. PC board is located at lower-left with the power jack just below that. The speaker is to the left of the video tube with the partially mounted pot above.

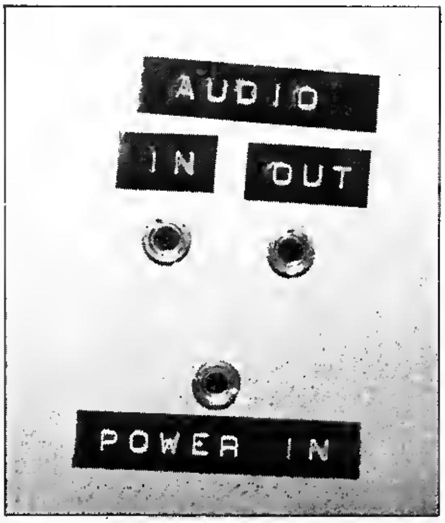


Photo 3. Close-up of Labeled Jacks

1—5K-ohm potentiometer	271-1714	$1.09
1—Potentiometer switch	271-1740	$0.69
1—Amplifier 200mW	277-1008	$11.95
1—Knob (sold two per pack)	274-414	$1.39

1—Phone jack. Size is dependent on power input plug. Check with your local RS store for the proper jack. If you use a 9-volt battery instead of the power supply you can get a battery clip instead of the jack. Note the installation instructions for the particulars.

Table 1. The parts listed are Radio Shack, but may be substituted with any brand.

1—1/4 inch nut driver to remove rear of video
Assorted drill bits ranging from very fine to 1/2 inch
Hand drill or locking pliers
1—Small common screwdriver
1—Very small cross point screwdriver
Light gauge wire (four colors): Shielded wire is not necessary
Pencil-style soldering iron (low watt)
Solder

Table 2. Tool List

casing so the jacks are pointing down with one jack centered over the pilot hole. Mark the location of the center of the second jack and drill that pilot hole. Enlarge the holes until the jacks fit through. Do not install the PC board yet.

If you are going to use an external power supply, decide on a location for the jack that will prevent confusion between jacks. I placed mine centered below the PC jacks (Photos 2 and 3). Drill this hole as the others. Be certain it is not too close to the PC board.

Now select the location for the speaker and drill four holes from the inside for sound to pass through. Inside the CRT, just below the installed pot, is a good location for the speaker (Photo 2). The holes can be covered in front with a black or silver grating (Photo 1). Do not mount the speaker at this time.

Once you are certain everything fits, remove the parts and prepare to connect the pot. Set the PC board so its soldered connections are facing you (Fig. 1). Solder one of the red switch wires to the first point. Solder the other red switch wire to the fifth point. Make sure the installed pot is in the off position. If you want, carefully bend back the switch contact to prevent accidentally turning the power on. Solder the remaining three wires from the remote pot to their corresponding points on the installed pot.

Now for the test. Connect power and audio input, and run a sound program. Turn the remote pot on and increase the volume. If the volume starts out loud and decreases or stays at one level, switch the second and fourth wires. The center wire should always be the center wire. If all checks out, install the system in the CRT casing and put on your finishing touches, such as jack labels.

That's all there is to it. Now you have sound in your unit where it belongs.

Oh, Something Else

People have asked how to get clear photos from the video display (Photo 1). The procedure uses a flash and untimed shutter speed.

Set the camera on a tripod angled

against flash rebounding on the CRT. Shoot as you would a normal flash shot, but hold the shutter open for 10, 20, and 30 seconds. The flash exposes the casing while the extended exposure burns in the screen contents without significantly darkening the rest of the photo.

Use a cable release when taking long exposures to prevent accidental movement.

I also use a green screen and a Radio Shack anti-glare screen. The screens prevent room light reflection. ■

Richard McGarvey is a police officer with a degree in psychology. He lives at 221 Hirschfield Drive, Williamsville, NY 14221.

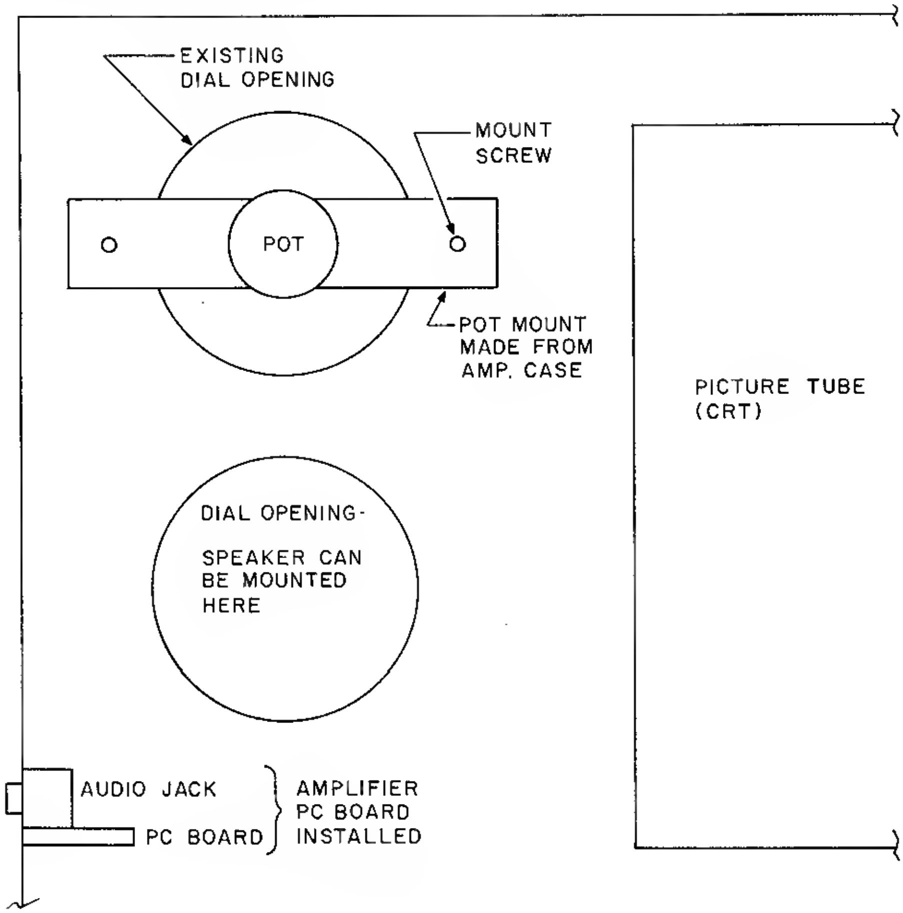


Fig. 2. CRT Casing with Back Removed

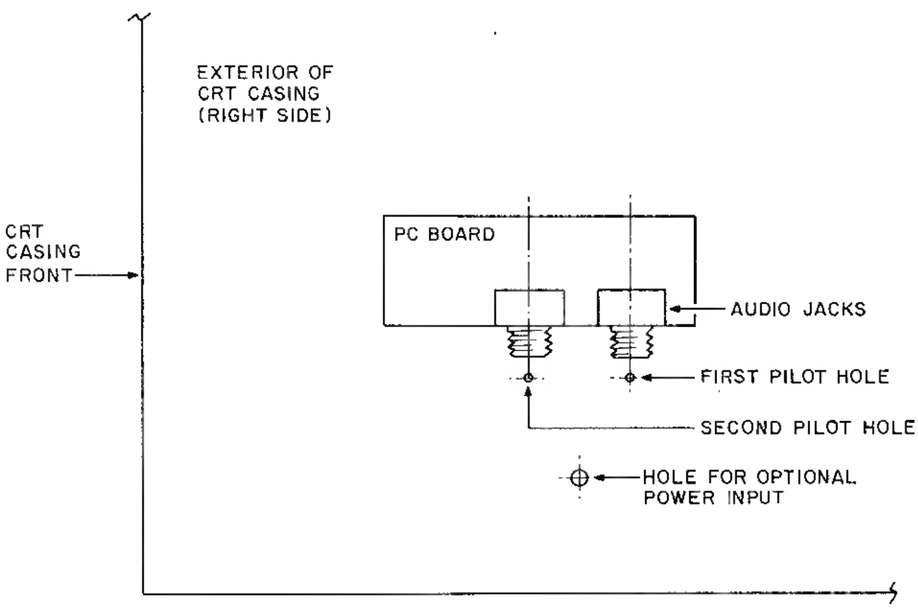


Fig. 3. CRT Casing Right Side Exterior

✓See List of Advertisers on Page 563

Android Picture Gallery

by M. K. Cook

Here's a new game for you! It seems that there are no humanoids left on the planet Rehabul. Who will take over dusting the art gallery?

Long ago the planet of Rehabul was inhabited by a humanoid species that, as it grew in sophistication, developed androids to help with the more mundane tasks. But once they had enough androids for the mundane tasks, a lot of android engineers were out of work.

This so alarmed the government that it gave large grants to the industry to keep it going and help it develop even better andriods to perform more interesting tasks.

Then, as more and more of the civilization's tasks were taken over by the andriods the humanoid species went into a decline. Robbed of all the interesting tasks, they lost their zest for life and did nothing. So a million years of evolution came to naught, and the humanoids became extinct.

Unfortunately, nobody notified the androids, who continued mechanically about their daily duties: composing sonnets, acting in plays, and painting. Where there are painters there must be galleries and exhibitions, the National Android Gallery being one of the most famous.

As a legacy of the long-since extinct humanoid species, things are not always as simple as they could be. An example is the anti-android duster used to clean the frames for the next exhibition. The duster removes the old android picture from the frame but has the annoying side-effect of reversing the nearby frames. The curator of the android gallery, therefore, has a puzzling task when he comes to clear the gallery to make way for the next exhibition.

Also, all the android portraits must be in frames, not suspended in space as the duster sometimes leaves them, before a new exhibition can be opened. The problematic duster means an extended training period is needed for android curators. To save on dusters, a computer simulation was written and has been mysteriously transferred to

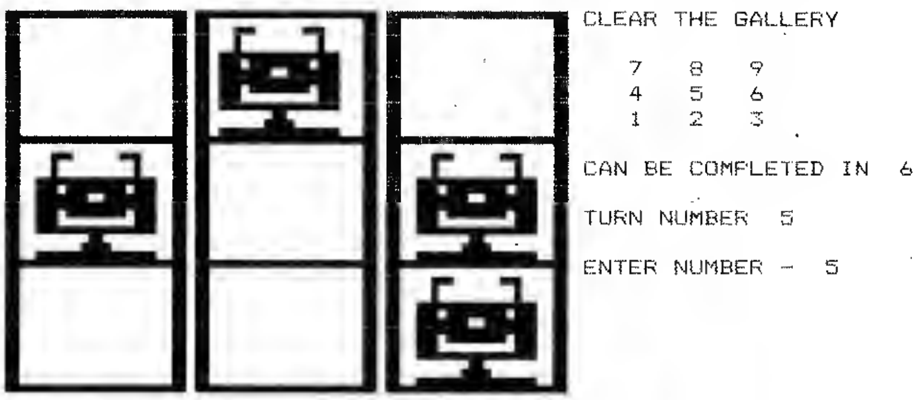


Figure 1

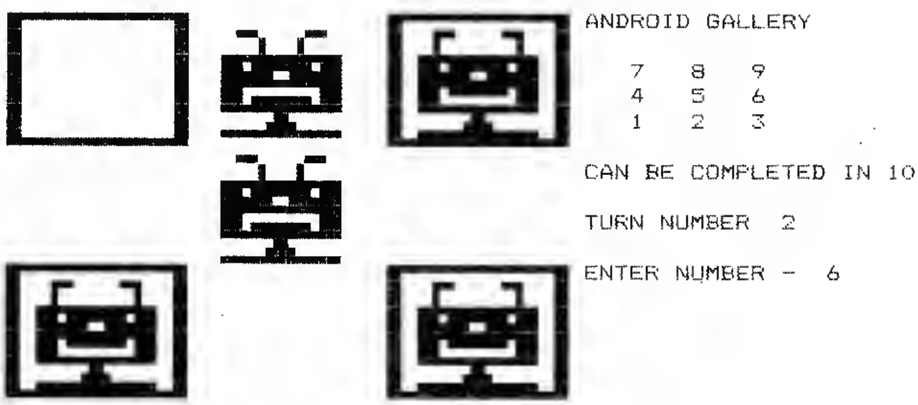


Figure 2

> ***The Key Box***
>
> **Model I or III**
> **16K, 32K RAM**
> **Cassette or Disk Basic**

earth via a crossed telephone line. So, you have the opportunity to train as an android curator as you play Android Gallery.

Android Gallery was written for a TRS-80 Model I computer using cassette or Disk Basic. It is two games in one—the clearing of the gallery and the setting up of an exhibition. In either mode you can let the computer set up your starting positions, in which case you are told the minimum number of duster applications needed to complete the task. Or you can set up the positions yourself.

To use the duster on a frame, press the key corresponding to that frame. It is the same layout as the numeric keyboard, but you can only use the duster on a frame, or a space, that is occupied with an android portrait.

Once you've sorted this out, you are left with the puzzle of organizing the gallery. The appearance and disappearance of the androids is accompanied by audio tones whose frequency corresponds to the frame number and duration, depending on whether the android portrait is appearing or disappearing. This allows the more adventurous reader to play the game blindfolded. (You could create a position where you could not possibly win, in which case you will be told.)

You might like to work out exactly how the program works and add more starting positions, not forgetting to include the minimum number of turns in which the puzzle can be completed. You might also like to tackle a program to solve the puzzle.

To test the program, enter the following initial conditions: For the simple game, set the Androids at 2, 5, 7, 8, 9 and play 5, 7, 9 to win.

For the full game, set the Androids at 4, 5, 6 and set frames at 1, 2, 3, 4, 6, 7, 8, 9; then play 6, 4, 5 to win.

Only try this if you're having trouble getting the program entered, since it might tell you something about the operation of the anti-android duster.

To adjust the program for Disk Basic, change the value in line 50. The accompanying diagrams are screen dumps of the game made by an MX-80 printer. One illustrates the simple game Clear the Gallery, and the other, the more complex, full game of Android Gallery. ■

M. K. Cook's address is 8 Fairhill, Helmshore Rossendale, Lancaster, BB4 45Y, England.

Program Listing

```
5 'ANDROID PICTURE GALLERY BY M.K.COOK G8HBR
10 CLS
20 CLEAR 1000
30 DEFINT A-Z
40 DIM PA(9,4),FG$(1,31),SG$(15)
50 BA=0 '0 FOR DISK BASIC 1 FOR LEVEL ii
60 RANDOM
70 FOR A=1 TO 9
80 READ FT(A)
90 NEXT
100 PRINTCHR$(23):PRINT:PRINTSTRING$(32,42);
110 FOR A=1 TO 9
120 READ RT(A)
130 NEXT
140 PRINT"**  ANDROID PICTURE GALLERY   **";
150 PRINTSTRING$(32,42)
160 PRINT:PRINT:PRINT"DO YOU WANT THE RULES ? ";
170 FOR A=1 TO 31
180 READ FG$(0,A),FG$(1,A)
190 NEXT
200 FOR A=1 TO 15
210 READ SG$(A)
220 NEXT
230 A$=INKEY$:IF A$="" THEN 230
240 PRINTA$
250 IF A$<>"Y" THEN 390
260 PRINTCHR$(28)
270 PRINT@256,
280 PRINT"YOU ARE THE CURATOR OF THE ANDROID GALLERY"
290 PRINT"IN THE SIMPLE GAME YOU HAVE TO - "
300 PRINT"CLEAR THE GALLERY READY FOR THE EXHIBITION"
310 PRINT"YOU DO THIS BY USING YOUR ANTI-ANDROID DUSTER"
320 PRINT"THIS WILL CLEAR AN OCCUPIED FRAME BUT UNFORTUNATELY"
330 PRINT"FRAMES NEAR BY WILL BE REVERSED"
340 PRINT"CORNER - CENTER - MIDDLE OF ROW   ----- EACH PRODUCE A
"
350 PRINT"DIFFERENT (BUT CONSISTENT) PATTERN OF CHANGE AND SOUND
."
360 PRINT"IN THE FULL GAME YOU HAVE TO GET ALL THE GIVEN FRAMES
OCCUPIED"
370 PRINT"BY HAPPY ANDROIDS (THEY'RE NOT HAPPY OUT OF A FRAME)."

380 PRINT"TYPING A 'Q' WILL QUIT THE CURRENT GAME AND START A NE
W ONE"
390 PRINT"TYPE S FOR THE SIMPLE GAME ":PRINT"ANY KEY FOR THE FUL
L VERSION ";
400 G$=INKEY$:IF G$="" THEN 400
410 PRINTG$
420 IF A$="Y" THEN 460
430 PRINT"DO YOU WANT TO ENTER YOUR OWN ":PRINT"STARTING POSITIO
NS ? ";
440 DA$=INKEY$:IF DA$="" THEN 440
450 PRINTDA$;
460 'SET UP FACE STRING
470 PRINT"  OK"
480 FOR N=0 TO 7
490 F$(N)=""
500 FOR N1=1 TO 8
510 READ T
520 F$(N)=F$(N)+CHR$(T)
530 NEXT N1,N
540 FOR N=1 TO 9
550 FOR N1=1 TO 4
560 READ PA(N,N1)
570 NEXT N1,N
580 BL$=STRING$(35,"A")
590 A=PEEK(VARPTR(BL$)+2):B=PEEK(VARPTR(BL$)+1)
600 AD1=A*256+B
610 IF AD1>32768 THEN C9=AD1-65536 ELSE C9=AD1
620 IF BA=0 THEN DEFUSR1=C9:CMD"T" ELSE POKE 16527,A:POKE 16526,
B
630 FOR A=C9 TO C9+29
640 READ T: POKE (A),T
650 NEXT
660 CLS
670 IF DA$<>"Y" THEN 860
680 FOR N=1 TO 9:BC(N)=0:NEXT
690 PRINTCHR$(23);"TYPE IN ANDROID POSITIONS"
700 PRINT"TYPE X WHEN FINISHED"
710 PRINT:PRINT:PRINT"ANDROIDS AT -";
720 GOSUB 2130
730 IF A$="X" THEN 750
740 BC(N)=1: GOTO 720
750 FOR N=1 TO 9
```

Listing continues

Listing continued

```
760 BF(N)=1
770 NEXT N
780 CLS
790 IF G$="S" THEN 1090
800 FOR N=1 TO 9:BF(N)=0:NEXT N
810 PRINTCHR$(23);"ENTER FRAME POSITIONS":PRINT"TYPE X WHEN FINI
SHED":PRINT
820 PRINT"FRAMES ARE AT - ";
830 GOSUB 2130
840 IF A$="X" THEN CLS:GOTO 1090
850 BF(N)=1:GOTO 830
860 NA=1:NF=0
870 FOR N=1 TO 9
880 BC(N)=0:BF(N)=0
890 NEXT
900 IF G$="S" THEN A$=SG$(RND(15)): TT$=RIGHT$(A$,2): GOSUB 2080
 : GOSUB 2200 : GOTO 1010
910 C=RND(31)
920 A$=FG$(1,C)
930 TT$=RIGHT$(A$,2)
940 GOSUB 2080
950 FOR N=1 TO 9
960 BF(N)=BC(N)
970 NEXT
980 FOR A=1 TO 9:BC(A)=0:NEXT
990 A$=FG$(0,C)+"  "
1000 GOSUB 2000
1010 FOR N=1 TO RND(4)
1020 FOR A=1 TO 9
1030 TF(RT(A))=BF(A):TC(RT(A))=BC(A)
1040 NEXT
1050 FOR A=1 TO 9
1060 BC(A)=TC(A):BF(A)=TF(A)
1070 NEXT
1080 NEXT
1090 TN=0
1100 FOR F=1 TO 9
1110 IF BF(F)=1 THEN GOSUB 1740
1120 NEXT F
1130 FOR F=1 TO 9
1140 IF BC(F)=1 THEN GOSUB 1840
1150 NEXT F
1160 P=38
1170 IF G$="S" THEN PRINT@P,"CLEAR THE GALLERY"; ELSE PRINT@P,"A
NDROID GALLERY";
1180 P=P+128
1190 PRINT@P,"    7    8    9 ";:P=P+64
1200 PRINT@P,"    4    5    6 ";:P=P+64
1210 PRINT@P,"    1    2    3 ";:P=P+128
1220 IF DA$<>"Y" THEN PRINT@P,"CAN BE COMPLETED IN ";TT$;
1230 P=P+120
1240 PRINT@P,"TURN NUMBER ";TN;:P=P+128
1250 PRINT@P,"ENTER NUMBER - ";:P=P+64
1260 A$=INKEY$:IF A$="" THEN 1260
1270 TN=TN+1
1280 IF A$="Q" THEN 660
1290 F=VAL(A$)
1300 PRINT@694,A$;
```

Listing continues

Listing continued

```
1310 PRINT@562,TN;
1320 IF F>6 THEN F=F-6 : GOTO 1340
1330 IF F<4 THEN F=F+6
1340 IF BC(F)=0 THEN 1260
1350 C=F
1360 BC(F)=0:GOSUB 1980
1370 FOR N2=1 TO 4
1380 F=PA(C,N2)
1390 IF F=0 THEN 1410
1400 IF BC(F)=0 THEN BC(F)=1:GOSUB 1840 ELSE BC(F)=0:GOSUB 1980
1410 NEXT N2
1420 ET=1:AC=1
1430 FOR N=1 TO 9
1440 IF BC(N)<>BF(N) THEN ET=0
1450 IF BC(N)=1 THEN AC=0
1460 NEXT N
1470 IF AC=1 AND G$<>"S" THEN 1590
1480 IF AC=1 AND G$="S" THEN 1500
1490 IF ET=0 THEN 1260
1500 PRINT@806,"THAT'S IT !! ";
1510 PRINT@934,"HIT KEY FOR NEW GAME";
1520 FOR A=9 TO 1 STEP -1
1530 POKE(C9+14),FT(A)
1540 IF BA=0 THEN X=USR1(150) ELSE X=USR(150)
1550 FOR A1=1 TO 30:NEXT
1560 NEXT
1570 A$=INKEY$:IF A$="" THEN 1570
1580 GOTO 660
1590 PRINT@806,"YOU CAN'T WIN NOW";
1600 GOTO 1510
1610 TC=F:TR=1
1620 IF TC<4 THEN 1650
1630 TC=TC-3:TR=TR+1
1640 GOTO 1620
1650 TC=TC-1:TR=TR-1
1660 RETURN
1670 X=(12*TC)+1
1680 IF TC=2  THEN X=X+1
1690 Y=64*(TR*5)+1
1700 L=X+Y
1710 X=(TC*24)+12:Y=TR*15
1720 FP=POINT(X,Y)[2
1730 RETURN
1740 GOSUB 1610
1750 X1=TC*25:X2=TC*25+22
1760 FOR Y=TR*15 TO TR*15+15
1770 SET(X1,Y):SET(X1+1,Y):SET(X2,Y):SET(X2+1,Y)
1780 NEXT Y
1790 Y1=TR*15:Y2=TR*15+15
1800 FOR X=TC*25 TO TC*25+22
1810 SET(X,Y1):SET(X,Y2)
1820 NEXT X
1830 RETURN
1840 GOSUB 1610
1850 GOSUB1670
1860 IF FP=1 THEN PRINT@L,F$(0);ELSE PRINT@L,F$(1);
1870 L=L+64
1880 IF BF(F)=1 THEN PRINT@L,F$(2);ELSE PRINT@L,F$(7);
1890 L=L+64
1900 PRINT@L,F$(3);
1910 L=L+64
1920 IF BF(F)=1 THEN PRINT@L,F$(4);ELSE PRINT@L,F$(6);
1930 L=L+64
1940 PRINT@L,F$(5);
1950 POKE(C9+14),FT(F)
1960 IF BA=0 THEN X=USR1(100) ELSE X=USR(100)
1970 RETURN
1980 GOSUB 1610
1990 GOSUB 1670
2000 IF FP=1 THEN PRINT@L,STRING$(8,131); ELSE PRINT@L,STRING$(8
,128);
2010 FOR N=1 TO 4
2020 L=L+64
2030 PRINT@L,STRING$(8,128);
2040 NEXT N
2050 POKE(C9+14),FT(F)
2060 IF BA=0 THEN X=USR1(200) ELSE X=USR(200)
2070 RETURN
2080 L=LEN(A$)-2
2090 FOR A=1 TO L
2100 BC(VAL(MID$(A$,A,1)))=1
2110 NEXT
2120 RETURN
2130 A$=INKEY$:IF A$="" THEN 2130
2140 IF A$="X" THEN RETURN
2150 N=VAL(A$)
2160 IF N>6 THEN N=N-6 : GOTO 2180
2170 IF N<4 THEN N=N+6
2180 PRINTA$;",";
2190 RETURN
2200 FOR A=1 TO 9
2210 BF(A)=1
2220 NEXT
2230 RETURN
2240 DATA 60,70,80,90,100,110,120,130,140,3,6,9,2,5,8,1,4,7
2250 DATA 1,2345678913,1346,789 5,25,12345610,25,134678911
2260 DATA 3,159 3,134679,124579 6,124,3678910,5,1234678911,456,1
2310,1379,5 9,9,1 8,12346789,5 5,2,5 8,12358,12346789 7
2270 DATA 5,2 4,5,1379 9,2,1345678911,5,46 7,258,2468 7,19,1 6,8
,1238 7,1,36789 0,124,3678910,19,37 0,5,19 7,5,456 8,258,134679
9
2280 DATA 19,137910,258,12346789 3,357,159 0,1,37 4
2290 DATA 13579 9,1 6,13789 5,2468 4,135 5,1234678 4,138 5
2300 DATA 2 9,137 4,5 5,123456789 5,159 3,37 6,1289 8,12358 3
2310 DATA 131,179,147,131,131,163,179,131
2320 DATA 128,176,144,128,128,160,176,128
2330 DATA 176,181,176,176,176,176,186,176
2340 DATA 191,189,191,183,187,191,190,191
2350 DATA 191,181,179,179,179,179,186,191
2360 DATA 176,176,176,190,189,176,176,176
2370 DATA 191,181,180,188,188,180,186,191
2380 DATA 176,176,186,176,176,181,176,176
2390 DATA 2,4,5,0,1,3,0,0,2,5,6,0
2400 DATA 0,1,7,0,2,8,4,6,3,9,0,0
2410 DATA 4,5,8,0,7,9,0,0,5,6,0,0
2420 DATA 205,127,10,14,1,219,255,203,119,32,2,203,217,6
2430 DATA 100,0,16,253,121,238,3
2440 DATA 211,255,79,43,124,181,32,240,201
```

100% NEW

80's 3 in '83!

And to celebrate we've published a special

ANNIVERSARY ISSUE,

containing

* all new articles & programs.
* over 500 pages of the same high 80 MICRO quality.
* dozens of games and programs for your computer.
* exclusive articles by well-known authors.
* the most in-depth coverage of the Color Computer yet.
* tutorials on having peripherals and software.
* page after page of Christmas specials from 80's advertisers.
* a 3-year 80 MICRO index—annotated and cross-referenced.

BUY ONE FOR YOUR SELF!

JUST IN TIME FOR CHRISTMAS

PLUS

THE FIRST 3D STEREOSCOPIC COLOR COMPUTER GRAPHIC

BUY ONE FOR A FRIEND!

GREAT STOCKING STUFFERS!

This amazingly versatile Anniversary Issue can go anywhere* with you. Fits easily in your briefcase or your purse. Lies flat in your knapsack. Rolls up to fit snugly in your glove compartment. Stores on any bookshelf. Looks great on any coffee table.

You can also get the amazing Special Edition LOAD 80 Companion to the Anniversary Issue. All those games and programs, just waiting for you to load. No more keyboarding. No more debugging. No more hassles.

You'd expect special editions of this quality to cost you hundreds of dollars. (Well, maybe dozens.) But for a limited time only, you can get the 80 MICRO ANNIVERSARY ISSUE for only $5.95. That's right—$5.95. And the LOAD 80 Companion is just $9.95 for cassette, $19.95 for floppy disk.

Note: This is an extra issue and will not be included as part of your regular subscription. So call today and order your very own Anniversary Issue and LOAD 80 Companion. Or give them as gifts. Just fill in the attached reply cards and drop them in the mail. Or call toll free—1-800-258-5473.

That's
That number again
That's

1-800-258-5473

Call today!

Operators are standing by (during business hours, of course). MasterCard and Visa accepted. (American Express, too.)

Please allow 6–8 weeks for delivery.

* Use of this product in the shower is not recommended by the publisher.

Using DEFFN

by Ralph Rideout

The DEFFN statement lets you create your own functions. Using the DEFFN statement simplifies your programming and saves memory.

The DEFFN function is a powerful Disk Basic statement. The TRSDOS Disk Basic manual defines the DEFFN statement as "(It) lets you create your own implicit functions." Don't look for your tattered calculus text yet! An implicit function is a variable defined by an equation or set of equations. For example, consider the following equation:

f(X) = A + B

The variable X is defined as the sum of A and B. Since A and B can take on any values in determining the value of X, the equation is an implicit function of X (f[X]). If the above equation read X = 1 + 2, it is obvious that the value of X must be three. In this case, X is defined by the explicit function 1 + 2.

The expression following the DEFFN is implicit because it is in a general algebraic form. For example, the above equation in the DEFFN expression would appear as follows:

DEFFNX(A,B) = A + B

Here the X variable is defined as the sum of A and B. In mathematical terms, X is the dependent variable while A and B are the independent variables. Since A and B occur within the parentheses, they are called arguments. Once the DEFFN expression is coded within a program, it can be evoked (called) by the statement FNX(A,B).

The expression is evaluated based on the current values of the arguments (A and B) within the program at the moment the function is initiated. For example, if A and B were 1.24 and 2.34, respectively, the FNX(A,B) would return the value 3.58, the sum of the arguments. The variable names (A and B) used as arguments are not assigned exclusively to the function; you don't have to use the same variables in calling the DEFFN statement. To add the variables X2 and HH together using the DEFFN expression, simply insert them as the arguments. For example, the statement FNX(X2,HH) returns the sum of X2 and HH. The arguments serve only as a means of passing values to the expression for evaluation.

Using the DEFFN statement in numerical expressions can simplify your programming by eliminating repetitious equations, thereby saving precious memory. There are other ap-

```
TB$="TITLE NAME" : TB%=FNTB%(TB$) : PRINTTAB(TB%);TB$
TN$="TITLE NAME" : TN%=FNTB%(TN$) : PRINTTAB(TN%);TN$
TN$="TITLE NAME" : PRINTTAB(FNTB%(TN$));TN$
TN$="TITLE NAME" : PRINT@0+FNTB%(TN$),TN$
```

Figure 1

```
100 MN$="JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"
110 DEFFNDT$(M%,D%,Y%)=MID$(MN$,M%+((M%-1)*2),3)+CHR$(47)+STRING
$(3-LEN(STR$(D%)),48)+RIGHT$(STR$(D%),LEN(STR$(D%))-1)+CHR$(47)+
RIGHT$(STR$(Y%),2)
120 INPUT"ENTER MONTH NUMBER ";M%
130 IF M%<1 OR M%>12 THEN 120
140 INPUT"ENTER DAY NUMBER ";D%
150 IF D%<1 OR D%>31 THEN 140
160 INPUT"ENTER YEAR NUMBER ";Y%
170 IF Y%<10 OR Y%>=100 THEN 160
180 DATE$=FNDT$(M%,D%,Y%)
```

Program Listing

The Key Box

Disk Basic
Model I or III
16K RAM
One disk drive

plications of the DEFFN function. The statement can be used for string manipulation as well.

DEFFN Strings

My first application of the DEFFN statement for string operations was a rather simple one: centering titles on the video display. The following statement computes the print tab position at which a title is centered.

```
DEFFNTB%(X$) = 32 - LEN(X$)/2
```

In the above statement variable TB% is the integer tab position to be computed from the string variable X$ (the title). The LEN statement returns the length of the string. The length is then divided by two and subtracted from 32. For example, if a title has 40 characters, the above function returns a tab position of 12 where the title (when printed) is centered on the video screen. Any valid Basic string variable name can be passed to the function via the X$ argument.

There are several methods to center a title. I use the four samples shown in Fig. 1. All four samples produce the same result. The choice is a matter of individual style or specific application. The above example is only one of the many uses I've found for the DEFFN function in string manipulations. The extent of its uses depends on your specific program requirements, but the following example may give you better insight into its power.

Formatting a Date

Many programs require date entries as part of filing information. The DEFFN function lets you create a date format with little programming effort. This is very important if you are working with random direct access files where a proper format is required by a specific number of bytes for a disk buffer. A date format may be defined for eight bytes of storage: MM/DD/YY. This numerical date format is very popular. Oftentimes, however, you need to have a date in a format that includes the name of the month (AUG/05/82) instead of its numerical counterpart. Although this is a more readable format, entering the month name and slashes many times can become tedious. The Program Listing shows how the DEFFN function can solve the problem.

Variable MN$ (line 100) is assigned the first three letters of each month. The DEFFN statement in line 110 uses this string for converting the numerical month entry into its alphabetic name. The final date format we seek is MMM/DD/YY. The string operations defining the final date (DT$) are computed based on the desired format from left to right, starting with the month. The formatting is accomplished with the use of several important Basic string statements: MID$, CHR$, STRING$, LEN, STR$, and RIGHT$. In the DEFFN statement, I have chosen the arguments M% (integer month number), D% (integer day number), and Y% (integer year number last two digits).

Take note of the data-entry lines (120–170). I have inserted validation checks on the month number, day number, and year number that are entered. This is a precaution to avoid an error message. It is also a good programming technique.

To see how the DEFFN function works, let's take an example through the process. Let's choose the date 8/5/82 or AUG/05/82. Since the desired final date format is MMM/DD/YY, the first DEFFN statement operation converts the numerical month input to the appropriate month name. This is accomplished through the MID$ statement. Here, the month number (M%) must be used to calculate the proper character position in the MN$. The computation is done in the expression M% + ((M%-1)*2). For example, if the month number is eight (AUG) the result of the calculation would be 8 + ((8 - 1)*2) or 22. The 22nd position of MN$ is where the month letters AUG begin. The third argument of the MID$ statement (numeral 3) returns the three characters beginning with the 22nd position, namely AUG, the desired month name.

Once the month name has been evaluated, the next step is to get the slash mark separating the month name from the day number. The program concatenates (links together) the month letters with CHR$(47), the character code for a slash. The result is AUG/.

Since the desired format (MMM/DD/YY) requires a day number with two characters, a single digit, such as 5, must become 05. A two-digit day remains unchanged. The Basic statements STRING$, LEN, STR$, and RIGHT$ perform the day evaluation.

✓See List of Advertisers on Page 563

In the STRING$ statement the numerical day (D%) is converted to a string (STR$[D%]) so it can be manipulated. Then subtract the length of the day string (LEN[STR[D%]]) from three to determine whether to insert a zero before a single-digit day (05). The value 48 in the last STRING$ position is the character code for a zero.

Subtract the length of the day string from three because, under the conditions of the program, the maximum length of any string day number is three. All numbers in the computer have a space preceding the actual value that is reserved for a negative sign, if the number is negative. Therefore, a single-digit number is actually two characters long when converted to a string by the STR$ statement, while a two-digit number is three characters long.

For the single-digit day, the STRING$ statement returns one zero. For the two-digit day, the number of zeros returned is zero or a null string, so no zeros will be placed before the day string.

The next statement, RIGHT$[STR$[D%],LEN[STR$[D%]] – 1], links the day number to the date. The statement returns a one-character string of a single-digit day and a two-character string of a two-digit day, beginning with the rightmost character. Starting at the right character position eliminates the first left position, which is reserved for the sign. A slash character (CHR$[47]) is then added before the year is concatenated. The date format at this point is AUG/05/.

The last statement converts the year number to a string and links the last two characters from the right (the year number) to the date. As stated above, the right characters are taken because the first blank position is reserved for the sign. After the year is added the final result is AUG/05/82.

Line 180 is where the DEFFN is evoked, passing the numerical date entries (M%, D%, and Y%) to the function, and assigning the formatted date (MMM/DD/YY) to the string variable DATE$.

DEFFN Function Advantages

Although the date format DEFFN statement is complicated, it demonstrates what you can do with one line of code that would otherwise require many lines of program statements to achieve the same objective. In the demonstration program presented here, only three numerical entries are required for inputting a date. Without the DEFFN statement, the same date format took 15 lines of code in my old program. I saved nearly 500 bytes of memory!

Another important advantage is program documentation. Defining frequently used variable routines in the DEFFN statement allows you and others to follow your program's logic flow with greater understanding. This is important when you are writing programs for others. Written instructions are clearer because the DEFFN functions are almost self-explanatory. You need not use as many remark statements within the program, thus avoiding longer execution times. Using DEFFN functions also improves your programming technique and gives your final work a professional appearance. ■

Ralph Rideout (3101 Morningside Drive, Raleigh, NC 27601) is a self-employed chemist.

FAST
EFFICIENT
MACHINE LANGUAGE

COLOR COMPUTER Action Games

SEND FOR OUR LATEST FLYER. GET ON OUR MAILING LIST.

ARCADE GAMES

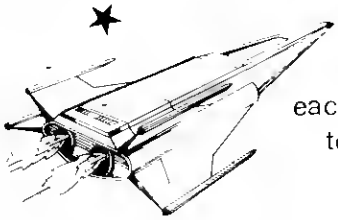
NEW!

Space Raiders

Not just another invaders type game. We think this one is the best—great action, great sound, you'll love it!!

CASSETTE (16K) $24.95
DISK (32K) $29.95

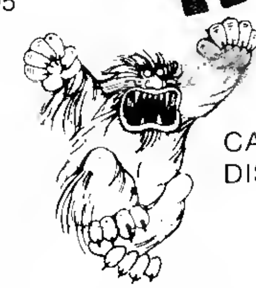
ASTRO BLAST



A new super hi-res space game. Wave after wave of alien attackers, each one a different and unique challenge to your skills.

CASSETTE (16K) . . . $24.95
DISC (32K) . . . $29.95

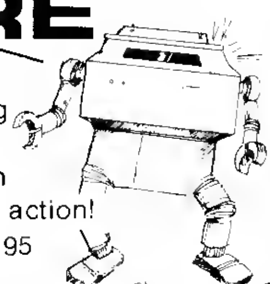
CAVE HUNTER



Outsmart the creatures that pursue you as you hunt for treasure in a maze of cave passages. Lots of colors and sounds!

CASSETTE (16K) . . . $24.95
DISC (32K) . . . $29.95

COLOR HAYWIRE



They're calling this one a "classic". You'll have hours of fast-paced fun zapping robots. Super hi-res action!

CASSETTE (16K) . . . $24.95
DISC (32K) . . . $29.95

"Finally . . . software worthy of the Color Computer" K.C. Westerly RI
"The ads and reviews were right. Great stuff!" R.S. Los Angeles CA

BUY DIRECT OR FROM YOUR FAVORITE DEALER

ADVENTURE GAMES

Calixto Island

A challenging puzzle with an occasional twist of humor. There's a treasure waiting to be discovered!

CASSETTE (16K) . . . $19.95

The Black Sanctum

For the player who enjoys suspense. . You'll encounter the forces of black magic in this spooky adventure.

CASSETTE (16K) . . . $19.95

MARK DATA PRODUCTS

23802 BARQUILLA, MISSION VIEJO, CA 92691 • (714) 768-1551

We pay shipping on all orders in the continental U.S. and Canada. Overseas add $3.00. California residents please add 6% sales tax. We are always looking for quality machine language programs. Contact us for details.

MASTER CHARGE OR VISA ACCEPTED

241

CATEGORIES

by Glenn Collura

If you're tired of fighting space creatures, Categories may be for you. You can play it without your 80, but this program adds spice.

Caution! This game may be habit forming. No, it's not the latest offering from Atari. In fact it's not an arcade-type game. You don't even need a computer to play this game.

What Is It?

Categories can be played by any number of people. All you need is a pencil and a piece of paper for each player. To start off, each player divides his paper into 20 squares as shown in Fig. 1. There are four columns and five rows. Each player then chooses a category (or categories, depending on the number of players) to enter onto the game sheet.

For example, you could use types of automobiles or the names of cities. Enter the categories on the left side of the game sheet corresponding to the five rows. After the categories have been chosen, four letters are chosen, one for each column on the game sheet. Once this is done you are ready to play the game.

How to Play

The object of the game is to find four words for each category, one word starting with each of the four different

The Key Box

Models I & III
16K or 32K RAM
Cassette or Disk Basic

Variable Name	Description
A$	Alphabet
B$	Categories
C1	Category 1
C2	Category 2
C3	Category 3
C4	Category 4
C5	Category 5
E$	Input Variable
I$	Input Variable
M	Minutes
N	One-Second Timer
P	Print Flag
Q	Print Loop Counter
R1	First Letter
R2	Second Letter
R3	Third Letter
R4	Fourth Letter
S	Seconds
T	Misc. For. . . Next Loop
X	Graphics Variable
Y	Graphics Variable
Z	Number of Game Sheets

Table 1. Variable List

	K	O	V	I
CANDIES				
TREES				
EXPLORERS				
TV SHOWS				
GAMES				

Figure 1

letters, in an allotted time period. If one of your categories was names of cities, you would enter the names of four cities each starting with the corresponding letter for that column. Refer to Fig. 2 for a completed game sheet.

Each player continues to fill in his game sheet until the five-minute time limit is reached, or until one player fills in his entire sheet before the five minutes is up. There will be times when there is no answer for a particular category/letter combination.

Scoring is very simple. Each correct answer scores one point. Any wrong answer or empty square counts as no point. The person receiving the highest score is the winner for that round. Since

"The object of the game is to find four words for each category, one word starting with each of the four different letters..."

there can be many correct answers for certain category/letter combinations, it is up to the players to check all answers and decide whether or not they are correct. The easiest way to do this is to start with the first category and the first letter and read the answer out loud. When all players have read their answers for that row, you can continue to the next, and so on through the entire game sheet. A game may consist of as many rounds as the players decide upon.

	B	D	O	K
CITIES	BOSTON	DENVER	OKLAHOMA CITY	KALAMAZOO
DISC JOCKEYS	BILLY BASS	GARY DEE		KID LEO
MUSICIANS	JEFF BECK	AL DIMEOLA	SHUGGIE OTIS	B.B. KING
HOBBIES	BASEBALL	DRAG RACING		KNITTING
CARTOON CHARACTERS	BULLWINKLE	DAFFY DUCK	ODIE COLONEY	KING LEONARDO

Figure 2

Why Use a Computer?

My wife and I have played this game for hours on end. It seems that after a few games we have a hard time thinking of categories to use. The microcomputer solves this problem quite easily.

I put as many categories as we could think of into the computer and then let the program generate five random categories and four random letters for each game. This also solves the problem of people choosing categories that they are more familiar with than are the other players. By randomly picking four letters of the alphabet, the computer prevents players from choosing letters that they know can be used with certain categories.

The computer also acts as a time keeper, notifying you when the time limit has expired. The last feature I added was the game-sheet printout option. The computer generates the game sheets for you if you have a printer. The program can be used without a printer, however. The main advantage of the printout option is that you can print any

number of game sheets at your convenience and have them available whenever you get the urge to play.

About the Program

The remark statements in the listing should point out any routines that you may want to modify. You may delete the remark statements without affecting program operation.

The first section of the program prints the instructions. The main program starts at line 1000. It checks for duplicates after each category and letter are chosen.

You may have to adjust line 2090 to ensure an accurate timer. If the five-minute timer seems slow, use a number less than 476. If it runs fast use a number greater than 476.

Line 4000 keeps the time display in an orderly format. The first statement in this line, PRINT@ 50,M": 0," makes sure that four minutes and nine seconds appears as 4:09 rather than 4: 9. The TRS-80 always leaves a leading blank

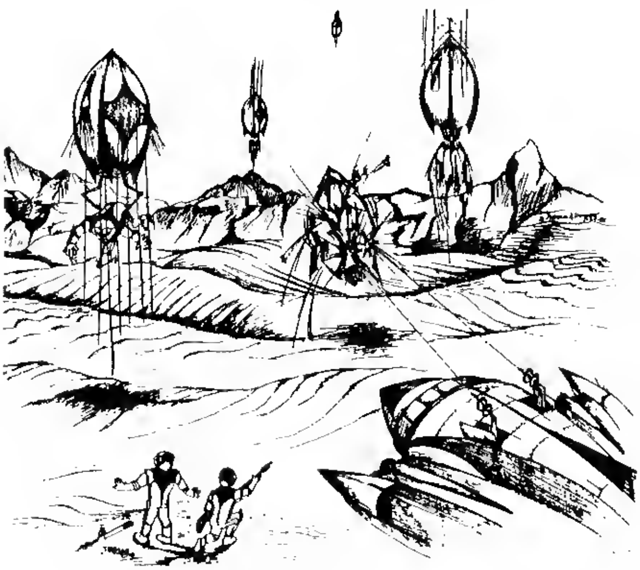
Program Listing

```
1 CLEAR100:DIMA$(25),B$(100)
2 RANDOM
3 DEFINTM,P,S,T,X,Y
10 REM ***** CATEGORIES *****
       *         BY         *
       ** GLENN COLLURA JR **
       ****** 03/01/80 ******
11 REM  VERSION 2.0  03/12/82
100 CLS:PRINT@20,"C A T E G O R I E S"
110 PRINT:PRINT:INPUT"WOULD YOU LIKE INSTRUCTIONS";I$
120 IFLEFT$(I$,1)="N"THEN600
150 CLS:PRINT:PRINT:PRINT"CATEGORIES IS A GAME PLAYED BY TWO OR
MORE PEOPLE.  IT IS PLAYEDON A PIECE OF PAPER DIVIDED INTO 20 WQ
UARES. YOU HAVE THE OPTIONOF LETTING YOUR TRS-80 PRINT THE GAME
SHEET FOR YOU.
155 PRINT:PRINT"IF YOU DO NOT HAVE A PSINTER, OR IF YOU WISH TO
MAKE UP YOUR OWNGAME SHEETS, FOLLOW MY INSTRUCTIONS."
160 PRINT:PRINT"YOUR SHEET OF PAPER SHOULD BE DIVIDED INTO 20 SQ
UARES AS SHOWN:"
170 PRINT:PRINT:INPUT"                   PRESS ENTER TO SEE DIAGRAM"
;E$
180 GOSUB5000
190 PRINT@256,"CATEGORY";:PRINT@384,"CATEGORY";:PRINT@576,"CATEG
ORY";
200 PRINT@704,"CATEGORY";:PRINT@832,"CATEGORY";
210 PRINT@78,"LETTER";:PRINT@89,"LETTER";:PRINT@100,"LETTER";:PR
INT@111,"LETTER";
220 PRINT@0,"PRESS ENTER TO CONTINUE";:INPUTE$;
230 CLS:PRINT:PRINT"FOR EACH GAME THE COMPUTER WILL GENERATE 5 R
ANDOM CATEGORIES ANDFOWR RANDOM LETTERS.  THE OBJECT OF THE GAME
 IS TO PUT A WORD INEACH COLUMN STARTING WITH THE LETTER FOR THA
T ROW.
240 PRINT:PRINT"LETS SAY FOR EXAMPLE THAT CATEGORY #1 IS SPORTS,
 AND YOUR FOUR  RANDOM LETTERS ARE   B     F     G      H .
250 PRINT:PRINT"THE TOP ROW OF YOUR SHEET WOULD LOOK SOMETHING L
IKE THIS:";
255 FORT=1TO450p:NEXT
260 FORX=2TO113:FORY=33TO40STEP7
270 SET(X,Y):NEXTY:NEXTX
280 FORX=25TO113STEP22:FORY=33TO40
290 SET(X,Y):NEXTY:NEXTX
295 PRINT@657,"B";:PRINT@667,"F";:PRINT@678,"G";:PRINT@690,"H";
300 PRINT@769,"SPORTS";:PRINT@782,"BASEBALL";:PRINT@793,"FOOTBAL
L";:PRINT@806,"GOLF";:PRINT@816,"HOCKEY";
310 PRINT@980,"PRESS ENTER TO CONTIN]E";:INPUTE$
320 CLS:PRINT"THE OBJECT OF THE GAME IS TO FILL IN ALL OF THE SQ
UARES. THERE  IS A FIVE MINUTE TIME LIMIT PER GAME. IF ANY PLAYE
R FILLS IN ALLTHE CATEGORIES BEFORE FIVE MINUTES, THE GAME IS OV
ER.
330 PRINT:PRINT"YOU SCORE ONE POINT FOR EACH CORRECT ANSWER.  IT
 IS UP TO YOU TOCHECK FOR CORRECT ANSWERS,  SINCE TLERE CAN BE M
ANY DIFFERENT   ANSWERS FOR EACH QUESTION. ANY EMPTY SQUARE COUN
TS AS A WRONG   ANSWER (NO POINTS). ";
335 PRINT"THERE MAY BE TIMES WHEN THERE IS NO ANSWER  FOR A CERT
QIN LETTER. (EX. A COLOR WITH THE LETTER  X.)
340 PRINT:PRINT"FOR YOUR CONVENIENCE,  THE TIME LEFT WILL BE,DIS
PLAYED IN THE   UPPER RIGHT HAND CORNER OF THE SCREEN.
350 PRINT"PRESS ' 1 ' TO REVIEW THE INSTRUCTIONS,
  OR  ' 2 ' TO START PLAYING."
355 I$=INKEY$:IFI$=""THEN355ELSEIFI$="1"THEN150
360 IFI$<>"2"THEN350
600 P=0:Z=0:INPUT"WOULD YOU LIKE TO PRINT GAME SHEETS";I$:IFLEFT
$(I$,1)="Y"THENP=1
610 IFP=1THENINPUT"HOW MANY SHEETS";Z
```

Listing continues

NEW BOOKS

NEW from Larry Ashmun!



His secrets are exposed in this new, super book.

ALIEN DEFENSE

The book contains:

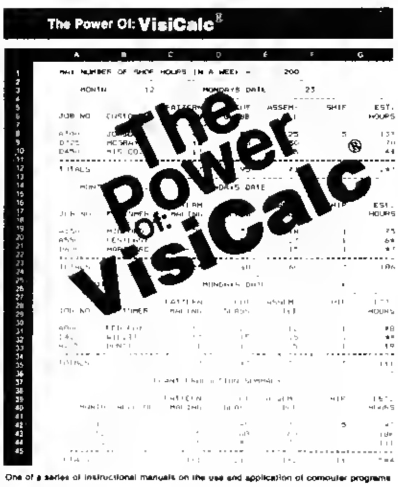
1. COMPLETE SOURCE CODE LISTING of Alien Defense (Mod III tape version).
2. Details of one approach to game writing.
3. Explains some of the more complicated routines.
4. You are informed of the concept behind the structure of the routines.

Never Before Has Anyone Sold a Book Like This!

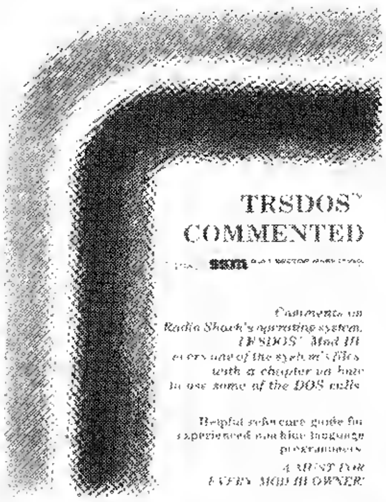
If you write in Machine language; if you have thought of writing a Machine language game; if you ever just wondered what a Machine language program looked like in its uncompiled form – This is Your Book!

Regularly $29.95 – INTRODUCTORY OFFER **$24.95**
Save $5.00 – Order Today
Offer Good thru Dec. 31, 1982

Other Fine Books Sold by Soft Sector Marketing, Inc.



$9.95



$29.95

$22.50

– DEALER INQUIRIES WELCOME –

86

SSM SOFT SECTOR MARKETING, INCORPORATED
P.O. Box 340 • Garden City, Michigan 48135
Order Line **800-521-6504**
Michigan Orders & Questions 313-425-4020

VISA · MasterCard · American Express

PAYMENT–payment accepted by charge, [illegible] or C.O.D. only, under the following co[illegible] **Charges** processed when shipped, usually wit[illegible] **Checks** delay shipping, pending 3 [illegible] **C.O.D.** orders are certified check or cash only, add $1.50. MI residents [illegible] sales tax.
SHIPPING & HANDLING – Shipping Charges: Send the larger [illegible] unless stipulated otherwise. Any order received without shipping and [illegible] **freight collect. Air Mail Shipping** outside of North America, pleas[illegible] $10.00. Overpayment will be refunded.

See List of Advertisers on Page 563

Listing continued

```
999 REM  MAIN ROUTINE
1ØØØ FORX=1TO25:READA$(X):NEXT
1ØØ9 REM  PICK FOUR LETTERS
1Ø1Ø R1=RND(25)
1Ø2Ø R2=RND(25):IFR2=R1THEN1Ø2Ø
1Ø3Ø R3=RND(25):IFR3=R2ORR3=R1THEN1Ø3Ø
1Ø35 R4=RND(25):IFR4=R3ORR4=R2ORR4=R1THEN1Ø35
1Ø37 GOSUB5ØØØ
1Ø4Ø PRINT@81,A$(R1);:PRINT@92,A$(R2);:PRINT@1Ø3,A$(R3);:PRINT@1
14,A$(R4);
1Ø45 PRINT@Ø,"HERE ARE YOUR LETTERS FOR THIS ROUND";
1Ø5Ø FORX=1TO88:READB$(X):NEXT
1Ø59 REM  PICK FIVE CATEGORIES
1Ø6Ø C1=RND(8Ø)
1Ø7Ø C2=RND(8Ø):IFC2=C1THEN1Ø7Ø
1Ø8Ø C3=RND(8Ø):IFC3=C2ORC3=C1THEN1Ø8Ø
1Ø9Ø C4=RND(8Ø):IFC4=C3ORC4=C2ORC4=C1THEN1Ø9Ø
1Ø95 C5=RND(8Ø):IFC5=C4ORC5=C3ORC5=C2ORC5=C1THEN1Ø95
1Ø97 FORT=1TO2ØØØ:NEXT
1Ø99 PRINT@Ø,"HERE ARE YOUR CATEGORIES            ";
2ØØØ PRINT@256,B$(C1);:PRINT@384,B$(C2);:PRINT@576,B$(C3);:PRINT
@7Ø4,B$(C4);:PRINT@832,B$(C5);
2ØØ5 IFP=1THENGOSUB6ØØØ
2Ø1Ø FORX=1TO3ØØØ:NEXT:PRINT@Ø,"PRESS ANY KEY TO START      ";
2Ø15 PRINT@5Ø,"5:ØØ";
2Ø2Ø I$=INKEY$:RESUMEI$=""THEN2Ø2Ø
2Ø59 REM  FIVE MINUTE TIMER
2Ø6Ø FORM=4TOØSTEP-1
2Ø7Ø FORS=59TOØSTEP-1
2Ø75 IFS<1ØGOTO4ØØØ
2Ø8Ø PRINT@5Ø,M":";S;
2Ø9Ø FORN=1TO476:NEXT: REM 1 SECOND TIMER
3ØØØ NEXTS
3Ø1Ø NEXTM
3Ø2Ø FORX=1TO1Ø:PRINT@Ø,"     TIME'S UP         ";
3Ø3Ø FORY=1TO6ØØ:NEXTY:PRINT@Ø,"                        ";
3Ø4Ø FORY=1TO3ØØzNEXTY:NEXTX
3Ø5Ø PRINT@Ø,"WOULD YOU LIKE TO PLAY ANOTHER GAME";:INPUTI$
3Ø6Ø IFLEFT$(I$,1)="Y"THENRESTORE:GOTO6ØØ
3999 END
4ØØØ PRINT@5Ø,M": Ø"RIGHT$(STR$(S),1);:GOTO2Ø9Ø
499Ø END
4999 REM  DRAW BOARD
5ØØØ CLS:FORX=25TO113STEP22:FORY=3TO44
5Ø1Ø SET(X,Y):NEXTY:NEXTX
5Ø2Ø FORX=2TO113:FORY=9TO44STEP7
5Ø3Ø SET(X,Y):NEXTY:NEXTXQDEFRETURN
5999 REM  PRINT GAME SHEETS
6ØØØ IFPEEK(14312)<>63PRINT@96Ø,"* * * PRINTER NOT READY * * *";
:GOTO2Ø1Ø
6ØØ5 FORQ=1TOZ
6Ø1Ø LPRINTSTRING$(71,"="):LPRINTTAB(22)"=      ";A$(R1);"      =
    ";A$(R2);"      =      ";A$(R3);"      =      ";A$(R4);"      =":
LPRINTINSTR(71,"="):GOSUB65ØØ
6Ø2Ø LPRINTB$(C1)TAB(22)"=            =            =            =
          =":GOSUB65ØØ:LPRINTSTRING$(71,"="):GOSUB65ØØ
6Ø3Ø LPRINTB$(C2)TAB(22)"=            =            =            =
          =":GOSUB65ØØ:LPRINTSTRING$(71,"=")CSAVEGOSUB65ØØ
6Ø4Ø LPRINTB$(C3)TAB(22)"=            =            =            =
          =":GOSUB65ØØ:LPRINTSTRING$(71,"="):GOSUB65ØØ
6Ø5Ø LPRINTB$(C4)TAB(22)"=            =            =            =
          =":GOSUB65ØØ:LPRINTSTRING$(71,"="):GOSUB65ØØ
6Ø6Ø LPRINTB$(C5)TAB(22)"=            =            =            =
          =":GOSUB65ØØ:LPRINTSTRING$(71,"="):LPRINTCHR$(1CONT8):N
EXTQ:RETURN
65ØØ FORT=1TO2:LPRINTTAB(22)"=            =            =  OUT
   =            =":NEXTT:RETURN
1ØØØØ DATA A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,Y,Z
1ØØ1Ø DATA ATHLETES,OCCUPATIONS,PRESIDENTS,COLOGNES,COLORS,TV SH
OWS,MOVIES,MAGAZINES,SONG TITLES,APPLIANCES,ACVORS,CITIES,MUSICA
L INSTRUMENTS,TOOLS,CAPITOL CITIES,COUNTRIES,STATES,PLANTS,FLOWE
RS,TREES
1ØØ2Ø DATA DOGS,FABRICS,ANIMALS,INVENTIONS,COMEDIANS,VEGETABLES,
FRUITS,CARS,MIXED DRINKS,TV DETECTIVES,HOBBIES,CANDIES,ICE CREAM
 FLAVORS,BOOK TITLES,AUTHORS,MUSICIANS,LQKES,FICTIONAL HEROES,SP
ORTS,INSECTS
1ØØ3Ø DATA ACTRESSES,COSMETICS,CIGARETTES,NATIONALITIES,BIRDS,AN
ATOMY,TRANSPORTATION,KITCHEN UTENSILS,GAMES,SNAKES,SEX SYMBOLS (
M&F),MICROCOMPUTERS,CLOTHES,DRUGS,MOUNTAINS,MUSICAL GROUPS,RIVER
S,TRAFFIC SIGNS,PASTRIES,DONUTS
1ØØ4Ø DATA AUTO PARTS,STREETS,CARTOON CHARACTERS,GEMS,NEWSCASTER
S,DEODORANTS,EXPLORERS,SPICES,BOYS NAMES,GIRLS NAMES,TOYS,DISC J
OCKEYS,FOOTBALL PLAYERS,BASEBALL PLAYERS,BOXERS,POETS,POLITICIAN
S,ELECTRONIC COMPONENTS,BRIDGES,MINERALS
```

for the sign of a number (+ or –). RIGHT$ and STR$ suppress the leading blank.

Line 6000 checks to see if the printer is ready and prints an error message if it is not. Lines 6005–6500 are the actual print routine for the game sheets.

Line 10000 contains the data for each letter of the alphabet. Lines 10010 and up are the data statements containing the categories.

Modifications

The routine to print the game sheets may be changed to suit your particular printer. Another modification would be an audible alert to signal the end of a game.

In its present form the program contains 80 categories. These may be changed if you have some categories that you would like to use instead of the ones already contained in the program. If you add any categories to the ones

"In its present form the program contains 80 categories."

already present, be sure to change line 1050 to correspond to the total number of categories you have. If you go over 100 categories, be sure to change the DIMB$ statement in line 1.

The method used to select the categories is the RND function. There have been numerous articles published dealing with generating "truly" random numbers. Any of these methods could be employed if you feel the selection of categories is not random enough.

I am not really sure when or where this game originated. I have heard that President John F. Kennedy used to play this game quite often when he was in office. I would appreciate hearing from any readers who could give me some background on this game.

This game is a lot of fun to play and has become quite addictive for more than a few people I know. If you miss your favorite tv show because you just had to play one more round of Categories, don't blame me! ■

Glenn Collura, a Field Service Engineer, enjoys fishing and playing the guitar. He can be reached at 9615 Seminole Trail, Streetsboro, OH 44240.

NOW!

The Ultimate in Mailing Packages

POSTMAN MASS MAILING SYSTEM

For the serious businessman who has as little as 100 name mailing list or 200,000 names, THERE IS ONLY ONE SYSTEM FOR YOU!

FEATURES OF THE NEW POSTMAN MASS MAILING SYSTEM

The Postman system (version 2) is an almost COMPLETE rewrite, rethink, redesign of the original POSTMAN. The many features of the new POSTMAN system are quickly outlined below:

MULTI-DRIVE – True multi-drive operation is possible. POSTMAN will search all drives for address files and connect them together into one large file for the duration of that session. Once POSTMAN has found the data files on the disks, the operator "sees" just ONE CONTIGUOUS sorted list of addresses. The operator does not need to tell POSTMAN when to "switch" drives or manually "swap" sections of the data file in and out of the computer's memory. This is the foremost among the list of features because of its relative uniqueness among mail list handlers written for the TRS-80.

LARGE LIST SUPPORT – The multi-drive operation allows the user to access data files on ALL configured drives CONCURRENTLY (at the SAME time) for truly large mailing lists. Files need not be sectioned into smaller "byte size" chunks to fit into memory.

HARD DISK SUPPORT – (HARD DISK POSTMAN only) The FULL utilization of the space and speed of the new hard disk drives is possible with POSTMAN. For example, a 7.5 megabyte drive can be configured to hold almost 60,000 labels. Multiple hard drives can be accessed CONCURRENTLY allowing 200,000++ entry mailing lists.

FORM LETTER CAPABILITY – With the purchase of the separate POSTRITE program, the user is provided with an easy to use form letter generator which will merge a generalized letter produced from a word processing system (i.e. LAZY WRITER, etc.), with the name and address information from the POSTMAN MASS MAILER data base. POSTWRITER allows the user to insert any field from a POSTMAN label entry anywhere in the letter.

MENU OPERATION – As you would in a restaurant, choose your dinner from a list (or MENU). POSTMAN will allow you to direct its actions by selecting from various menus that it will display. A complete discussion of each menu is presented in the manual.

INSERT – New names can be quickly added to your list at any time. The new addresses are placed into the file in their proper sorted order eliminating the need for a separate sort operation after entering a stack of new names. POSTMAN will allow the operator to enter a "batch" of labels without returning to the control menu between each label insertion, thus speeding entry and reducing the aggravation of extra menu control keystrokes.

DELETE – Names can be removed at any time when they are no longer needed.

EDIT – Information in any name entry can be quickly changed at will with "word processor" ease. A "transparent" cursor simply is moved to the label displayed on the computer screen and corrections are just typed over the existing label. If you happen to change a field which is also used as a sort key, POSTMAN will automatically move the changed label to its correct position in the list to maintain the sorted arrangement of the labels.

OVERLAY – When identical changes are needed on many addresses the OVERLAY feature can make them with one keystroke. The needed changes which are common to many labels are entered into the "overlay mask". When you wish to apply these common changes to any label, one command will do it.

SORT – Arrange your list in any alphabetic or numeric order. The ordering may use one or more fields to control the sort. A machine language heap sort assures fast execution. The sort need only be performed once, the sorted list will stay sorted through all subsequent insertions, deletions, and changes to existing labels. NO NEED to leave the POSTMAN program to use a separate program to sort your data. Your data is sorted quickly and after sort completion, POSTMAN is ready for your next command!

SPECIAL STREET ADDRESS SORT – For the user with many addresses on the same street, POSTMAN will sort your entries by the house NUMBER after grouping those on the same street together. Local city lists can be quickly sorted to aid post office dispatching.

PURGE – Unwanted duplicate addresses can be removed from your list automatically or under operator control.

SEARCH – Any address in your list can be quickly found with fast search and positioning commands. Three different types of searches are provided. A "fast" search which uses a hashing technique, a "selective sequential" search for labels with common fields, and "quick" positioning using the first or major sort field to get you into the general "ball park" of a label or sequence of labels.

LABEL PRINTING – One, a few or all addresses in your list can be printed on standard or non-standard label stock. Up to 6 labels across can be printed with a format YOU can easily control. TWO user definable 'ATTN' lines are provided for any use. Labels can be printed from many of POSTMAN's menus, search, edit, or during label insertion.

EFFICIENCY – POSTMAN is written in the machine's native language to gain the full advantage of the microcomputer's speed. Extensive use of program segmentation reduces the amount of use RAM needed to hold the program, allowing a greater number labels to be kept in core, resulting in faster operation. Little used routines need only be brought into memory when they are needed and once through with their task, release their space back to POSTMAN.

REPORT LISTINGS – A special program to produce columnar listings of address data from your label data base is provided. You can easily specify the information to be printed.

DATA DISK MERGING – Labels can be quickly transferred from one disk to another with the PSTMERGE program callable from the main POSTMAN SYSTEM menu. Source and destination drives needed not be separate drives, prompts to exchange diskettes if the same drive is used, are provided.

DATA DISK PREPARATION UTILITY – Provided with POSTMAN is the DPREP program which allows the user to prepare a floppy/hard disk for use with POSTMAN. This easy to use utility can be told to prepare any portion of the available space on a disk.

DATA INTEGRITY – All data transfers to the disk files are made using special write commands which instructs the operating system to check the validity of EACH write to the disk.

DATA GUARD™ – Is a special programming technique only offered by Soft Sector Marketing, Inc. If by chance your machine resets while writing information to the disk, you only lose the information that you were writing. Your files are always protected from the danger of losing all the work that you have put in that day. NO OTHER PROGRAM ON THE MARKET OFFERS THIS PROTECTION. If you reset with ANYBODY'S MAILING PACKAGE DURING WRITING you would destroy your ENTIRE data disk. We can't stop your machine from failing but we can protect your data.

Description of Label Record Fields:

Length	Name	Description
10	Code	User defined printable field
15	Last Name	Last name of addressee
15	First Name	First name of addressee
26	Company	Name of company
26	Address	Street address
15	City	City, township, village
5	State	State, province, territory
9	Zip	Zip code, zone, route
2	Data 1	User definable field
5	Data 2	User definable field

IDEAL SYSTEM

Mod III 48K 1-40 Track Drive • 2-80 Track Dual Headed Drives • Dosplus or LDOS Operating Systems

Gives space for over 11,000 names - 5 second average name insertion - time sorts all 11,000 names in less than 4 minutes

*Special version to work on Dosplus 4.0 Hard Disk operating system.

– Overview Available –

The POSTMAN system requires Mod I or Mod III, 48K, 2 disk drives minimum.

Standard Version	Standard Version with POSTWRITER form letter writer	For DOSPLUS Hard Disk 4.0 Operating System & Radio Shack's Hard Drive System
Only **$125.00**	**$175.00**	**$225.00**

✓87

– DEALER INQUIRIES WELCOME –

SSM SOFT SECTOR MARKETING, INCORPORATED

P.O. Box 340 • Garden City, Michigan 48135

Order Line **800-521-6504**

Michigan Orders & Questions 313-425-4020

VISA® MasterCard American Express

PAYMENT – payment accepted by charge, personal c[illegible] or C.O.D. only, under the following conditions. **Ch[illegible]** processed when shipped, usually within 48 hours. **[illegible] Checks** del[illegible] shipping, pend[illegible]g 3 weeks to cle[illegible] orders are certified check or cash only; add [illegible] MI residen[illegible]st add 4% s[illegible]

SHIPPING & HANDLING – Shipping Char[illegible] the l[illegible]unt, 2% [illegible] stipulated otherwise. Any order received witho[illegible] **collect Air Mail Shipping** outside of North [illegible] $10.00. Overpayment will be refunded.

✓See List of Advertisers on Page 563

Number Your Program Listings

by Joe Edwards

Here's a utility that will come in handy for you Basic programmers—a machine-code routine that numbers pages of multi-page listings.

This routine numbers pages of multi-page Basic program listings. I have used it with several utility programs; by relocating the program and changing the jump instruction in line 250 to a halt instruction, you could use it in smaller systems without disk. (The expansion interface or some type of parallel interface for the printer is still necessary.)

Program Listing

```
                00100 ;PAGE NUMBERING FOR BASIC (FILESPEC: PAGE/CMD)
                00110 ;BY JOE EDWARDS, 7130 BRONNER CIRCLE
                00120 ;LOUISVILLE, KY. -- AUGUST 1, 1981
                00130
                00140 ;FORM FEED (CHR$(12) IN BASIC) REINITIALIZES PROGRAM
                00150 ;WITHOUT RE-LOADING. RESET PAPER WITH PRINT HEAD
                00160 ;APPROXIMATELY SIX LINES DOWN FROM PERFORATION.
                00170 ;PAGE NUMBER 01 IS NOT NUMBERED.
                00180 ;THIS PROGRAM ALSO WORKS WITH NEWDOS EDTASM, DISASSEM,
                00190 ;AND RSM MONITOR
                00200
FD00            00210          ORG     0FD00H
FD00 2124FD     00220 START    LD      HL,PRNTR ;LOAD DRIVER ADR.
FD03 222640     00230          LD      (4026H),HL ;STORE DVR ADDRESS
FD06 CD0CFD     00240          CALL    INIZ     ;INITIALIZE COUNTERS
FD09 C32D40     00250          JP      402DH    ;RETURN TO DOS
FD0C 3E4F       00260 INIZ     LD      A,79D    ;SET CHARACTER COUNT
FD0E 3296FD     00270          LD      (CHCNT),A ;STORE IT
FD11 3E30       00280          LD      A,30H    ;1ST ASCII DIGIT OF NBR.
FD13 3297FD     00290          LD      (PGNBR1),A ;STORE IT
FD16 3E32       00300          LD      A,32H    ;2ND ASCII DIGIT OF NBR.
FD18 3298FD     00310          LD      (PGNBR2),A ;STORE IT
FD1B DD212540   00320          LD      IX,4025H ;INIZ DCB POINTER
FD1F DD360400   00330          LD      (IX+4),0 ;CLEAR LINE COUNTER
FD23 C9         00340          RET
                00350
FD24 ED7394FD   00360 PRNTR    LD      (STACK),SP ;SAVE STACK PNTR.
FD28 DD212540   00370          LD      IX,4025H;INIZ DCB POINTER
FD2C 79         00380          LD      A,C      ;RETURN IF 0
FD2D B7         00390          OR      A
FD2E C8         00400          RET     Z
FD2F FE0C       00410          CP      0CH      ;SEE IF FORM FEED
FD31 28D9       00420          JR      Z,INIZ   ;YES, REINITIALIZE
FD33 FE0D       00430          CP      0DH      ;CHECK FOR CR
FD35 282B       00440          JR      Z,CARRET;YES, DO IT
FD37 F5         00450 COUNT    PUSH    AF       ;SAVE ACCUM.
```

Listing continues

Here's the Plan

Using EDTASM, type the source code and dump the program to disk. Now, load the program in memory. When control returns to DOS, enter Basic. Use 65279 for memory size. With the printer on, set the print head six lines down from the top perforation. Now, run the following test program:

```
10 FOR I = 1 TO 70
20 LPRINT I
30 NEXT
40 END
```

The program should print digits 1–54 down the left margin of the page, skip to the top of the next page, print PAGE 02 in the upper right corner, and then digits 55–70 down the left margin.

To reinitialize the page counter without reloading from disk, type LPRINT CHR$(12). This feature is useful when running several copies of multi-page data. ■

Joe Edwards, 7130 Bronner Circle, Louisville, KY 40218, is retired. His hobbies are ham radio and flying.

The Key Box

Basic Level II
Model I
48K RAM
Disk System
Parallel Printer

See List of Advertisers on Page 563

Listing continued

```
FD38 2196FD     00460        LD      HL,CHCNT  ;GET CHAR. COUNT
FD3B 35         00470        DEC     (HL)      ;DECREMENT IT
FD3C 2003       00480        JR      NZ,COUNT1 ;NOT ZERO, JUMP
FD3E CD62FD     00490        CALL    CARRET    ;DO CAR. RETURN
FD41 F1         00500 COUNT1 POP     AF        ;RESTORE ACCUM.
FD42 1833       00510        JR      PRINT     ;PRINT CHARACTER
                00520
FD44 C5         00530 TAB    PUSH    BC        ;SAVE REGISTER
FD45 F5         00540        PUSH    AF        ;SAVE ACCUM.
FD46 3E0A       00550        LD      A,10D     ;SET COUNT
FD48 F5         00560 TAB1   PUSH    AF        ;SAVE COUNT
FD49 0E0D       00570        LD      C,0DH     ;LOAD CR CODE
FD4B CD24FD     00580        CALL    PRNTR     ;PRINT IT
FD4E F1         00590        POP     AF        ;GET COUNT
FD4F 3D         00600        DEC     A         ;DECREMENT IT
FD50 20F6       00610        JR      NZ,TAB1   ;LOOP IF NOT DONE
FD52 CD99FD     00620        CALL    PAGE      ;PRINT PAGE NO.
FD55 CD62FD     00630        CALL    CARRET    ;PRINT CR
FD58 CD62FD     00640        CALL    CARRET    ;PRINT CR
FD5B DD360400   00650        LD      (IX+4),0  ;CLEAR LINE CNTR.
FD5F F1         00660        POP     AF        ;RESTORE ACCUM.
FD60 C1         00670        POP     BC        ;RESTORE REGISTER
FD61 C9         00680        RET
                00690
FD62 3E4F       00700 CARRET LD      A,79D     ;SET CHAR. COUNT
FD64 3296FD     00710        LD      (CHCNT),A ;STORE IT
FD67 DD3404     00720        INC     (IX+4)    ;INCR. LINE CNT.
FD6A DD7E04     00730        LD      A,(IX+4)  ;COMPARE
FD6D FE37       00740        CP      55D       ;IF 55 LINES
FD6F 28D3       00750        JR      Z,TAB     ;DO VERTICAL TAB
FD71 3E0D       00760        LD      A,0DH     ;LOAD CR
FD73 CD77FD     00770        CALL    PRINT     ;PRINT IT
FD76 C9         00780        RET
                00790
FD77 4F         00800 PRINT  LD      C,A       ;SAVE A COPY
FD78 3AE837     00810 PRINT1 LD      A,(37E8H) ;CHECK LP STATUS
FD7B CB7F       00820        BIT     7,A       ;CHECK BUSY BIT
FD7D 20F9       00830        JR      NZ,PRINT1 ;LOOP IF HIGH
FD7F 79         00840        LD      A,C       ;GET A CHARACTER
FD80 32E837     00850        LD      (37E8H),A ;SEND IT TO PRINTER
FD83 79         00860        LD      A,C       ;GET CHARACTER AGAIN
FD84 CD3300     00870        CALL    33H       ;PUT ON SCREEN
FD87 3A4038     00880 BREAK  LD      A,(3840H) ;CHECK FOR BREAK KEY
FD8A FE04       00890        CP      4
FD8C 2801       00900        JR      Z,BREAK1
FD8E C9         00910        RET
FD8F ED7B94FD   00920 BREAK1 LD      SP,(STACK) ;RESTORE ORIG STACK
FD93 C9         00930        RET
                00940
0002            00950 STACK  DEFS    2         ;STACK PTR. STORAGE
FD96 4F         00960 CHCNT  DEFB    79D       ;CHAR. COUNTER
FD97 30         00970 PGNBR1 DEFB    30H       ;1ST DIGIT OF NUMBER
FD98 32         00980 PGNBR2 DEFB    32H       ;2ND DIGIT OF NUMBER
                00990
FD99 F5         01000 PAGE   PUSH    AF        ;SAVE ACCUM.
FD9A 3E43       01010        LD      A,67D     ;SET SPACE CNTR.
FD9C F5         01020 PAGE1  PUSH    AF        ;SAVE COUNT
FD9D 0E20       01030        LD      C,20H     ;LOAD SPACE CODE
FD9F CD24FD     01040        CALL    PRNTR     ;PRINT A SPACE
FDA2 F1         01050        POP     AF        ;GET COUNT BACK
FDA3 3D         01060        DEC     A         ;DECREMENT COUNT
FDA4 20F6       01070        JR      NZ,PAGE1  ;LOOP IF NOT ZERO
FDA6 F1         01080        POP     AF        ;RESTORE ACCUM.
FDA7 21D9FD     01090        LD      HL,STRING ;POINT TO TEXT
FDAA E5         01100 LOOP   PUSH    HL        ;SAVE ADDRESS
FDAB 4E         01110        LD      C,(HL)    ;LOAD A CHARACTER
FDAC CD24FD     01120        CALL    PRNTR     ;PRINT IT
FDAF E1         01130        POP     HL        ;RESTORE REGISTER
FDB0 23         01140        INC     HL        ;LOOK AT NEXT ADDRESS
FDB1 7E         01150        LD      A,(HL)    ;PUT CHAR. IN ACCUM.
FDB2 FE00       01160        CP      0         ;SEE IF ZERO
FDB4 20F4       01170        JR      NZ,LOOP   ;IF NOT, DO MORE
FDB6 3A97FD     01180        LD      A,(PGNBR1) ;LOAD 1ST DIGIT
FDB9 4F         01190        LD      C,A       ;PUT IT IN C REG.
FDBA CD24FD     01200        CALL    PRNTR     ;PRINT IT
FDBD 3A98FD     01210        LD      A,(PGNBR2) ;LOAD 2ND DIGIT
FDC0 4F         01220        LD      C,A       ;PUT IT IN C REG.
FDC1 CD24FD     01230        CALL    PRNTR     ;PRINT IT
FDC4 3A98FD     01240        LD      A,(PGNBR2) ;PUT 2ND DIGIT IN ACCUM.
FDC7 3C         01250        INC     A         ;INCREMENT IT
FDC8 FE3A       01260        CP      3AH       ;IS IT MORE THAN NINE
FDCA 2009       01270        JR      NZ,AHEAD  ;IF NOT, JUMP AHEAD
FDCC 3A97FD     01280        LD      A,(PGNBR1) ;LOAD 1ST DIGIT
FDCF 3C         01290        INC     A         ;INCREMENT IT
FDD0 3297FD     01300        LD      (PGNBR1),A ;STORE IT
FDD3 3E30       01310        LD      A,30H     ;LOAD NEW 2ND DIGIT
FDD5 3298FD     01320 AHEAD  LD      (PGNBR2),A ;STORE IT
FDD8 C9         01330        RET
FDD9 50         01340 STRING DEFM    'PAGE '
FDDE 00         01350        DEFB    00
FD00            01360        END     START
00000 TOTAL ERRORS
```

HARDWARE SPECIALS

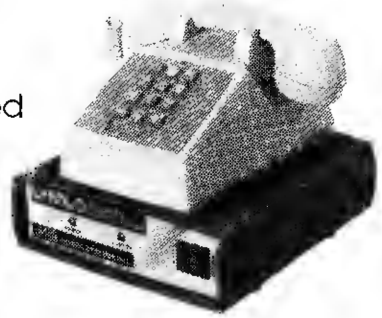
LYNX MODEM

Auto Answer - Auto Dial

Mod I & III
No RS-232
Board Required

Shipping
from Stock

$218.00



RIBBONS

MX-80 Replacement Cartridges
2 for $15.00
MX-100 Replacement Cartridges
2 for $20.00
Offer Good thru Dec. 31, 1982

ORCHESTRA 85/90

SUPER DEAL

Stereo music synthesizer with percussion for your Mod I or III.

Was $149.95

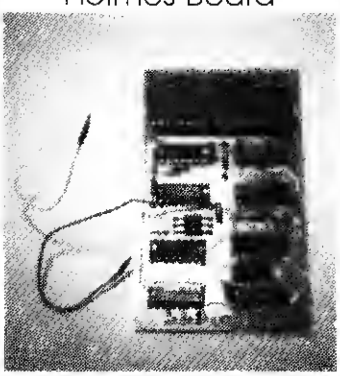
SPECIAL $99.95

Please state Mod I or II & Cass. or Disk version
Offer Good thru Dec. 31, 1982

Our Catalog is Now Available!

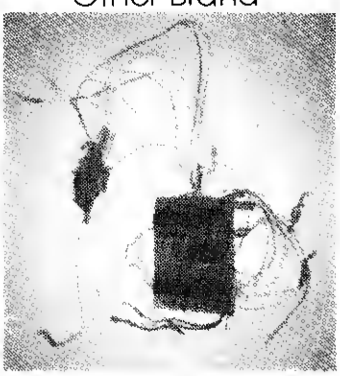
There Are Only 2 Ways to Increase the Processing Speed of Your Model III

Holmes Board



The Right Way

Other Brand



The Wrong Way

They both list for $99.95 assembled and Tested.
They both speed up your machine.

	Holmes Board	Other Brand
Installation Time	15 min.	4 hrs.
Requires Soldering	NO	YES
Requires Cutting Traces ...	NO	YES
Easily Removed if Logic Board Requires Repair ...	YES	NO

We Offer Only The Quality Product
But the Choice is Yours

SAVE $10.00

List $99.95
Only **$89.95**
Plus Shipping
Offer Good Until Dec. 31, 1982

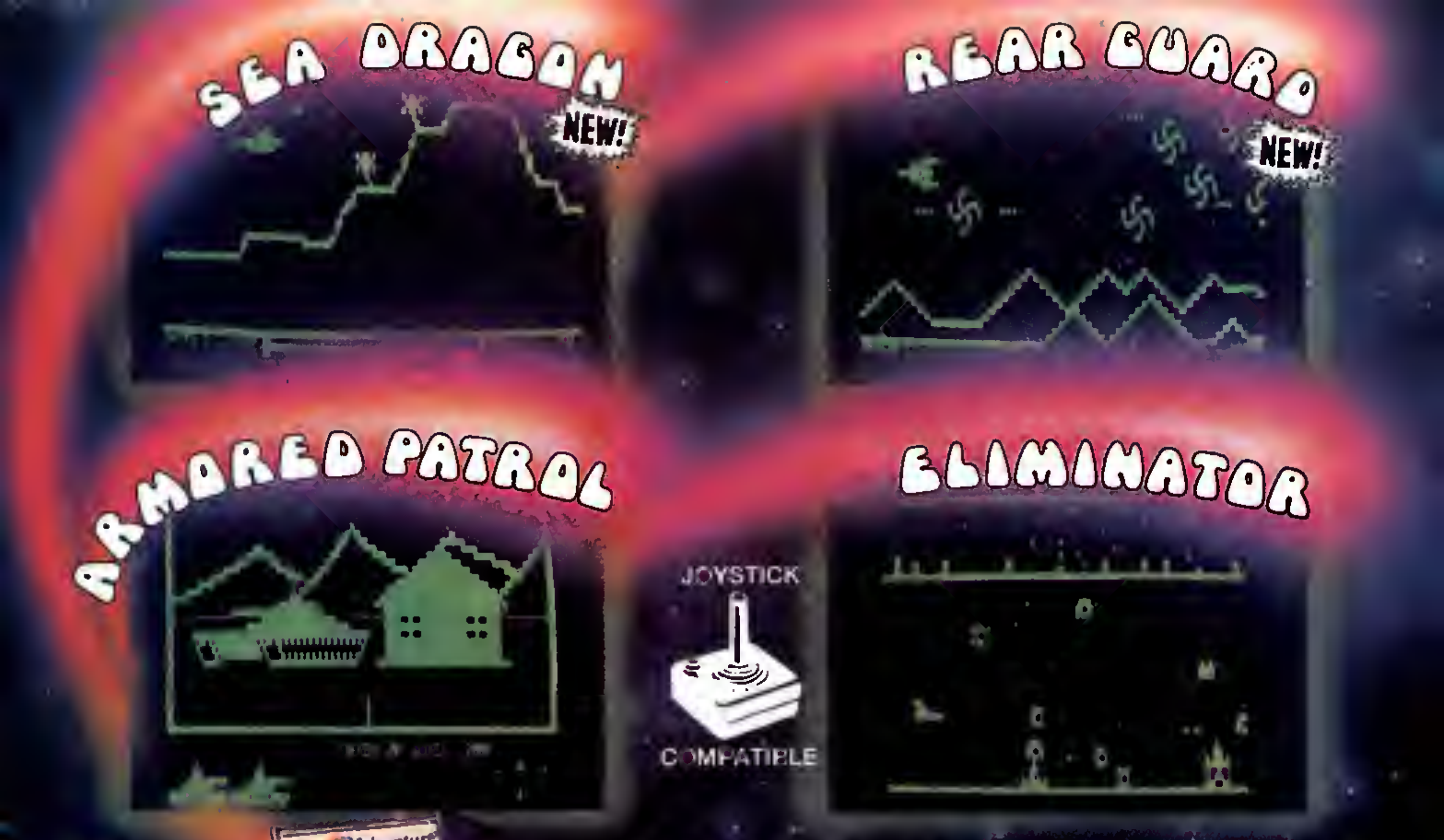
– DEALER INQUIRIES WELCOME –

89

SSM SOFT SECTOR MARKETING, INCORPORATED
P.O. Box 340 • Garden City, Michigan 48135
Order Line **800-521-6504**
Michigan Orders & Questions 313-425-4020

VISA MasterCard AMERICAN EXPRESS

PAYMENT– payment accepted by charge, personal c[illegible] or C.O.D. only under the following conditions. **Ch[illegible]** processed when shipped, usually within 48 hours. **[illegible] Checks** delay shipping, pending 3 weeks to cl[illegible] **C.O.D.** orders are certified check or cash only, add [illegible] MI residen[illegible] must add 4% [illegible]
SHIPPING & HANDLING – Shipping Charge[illegible] [illegible] the la[illegible]unt 2% [illegible] unless stipulated otherwise. Any order received witho[illegible] **freight collect Air Mail Shipping** outside of Nort[illegible] $10.00. Overpayment will be refunded.



YOUR TRS-80 ISN'T SUPPOSED TO DO THIS!

Wayne Westmoreland and Terry Gilman have done it AGAIN!

The reviews are saying what the fans have known all along: "The Eliminator" and "Armored Patrol" are nothing short of fantastic. Read for yourself.

Byte Magazine (on Armored Patrol): "A real super job with the graphics."

Byte Magazine (on The Eliminator): "The best thing to happen to the TRS-80 in a long time."

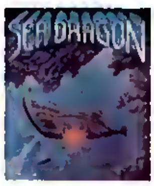
80 Microcomputing (on both): "This is state-of-the-art stuff."

80 U.S. (on The Eliminator): "The graphics are vastly superior to that of any game currently available ... it is slick, professional, and, without qualification, a blast."

And now, two new releases — and both are incredible arcade-style games. The TRS-80 is almost transformed into an arcade machine, performing Arcade Action Graphics(tm) like never before. Rear Guard and Sea Dragon are great fun and will provide you with hours of enjoyment. Both programs are available now, and all four are compatible with the Alpha Joystick!

ARMORED PATROL: You're in the command seat of an armored assault tank. Your mission: to search out and destroy an attack force of deadly tanks and lethal robots. Will your skill and advanced technology give you the edge to overcome the odds?
Armored Patrol Features: Simulated 3-D perspective, multiple player option, real time, sound and Arcade Action Graphics(tm) second to none.

THE ELIMINATOR: You're the pilot of The Eliminator, an attack class fighter that's squared off against hordes of attacking alien spaceships. It's your speed and weaponry versus their seemingly endless numbers. Can you possibly survive?
Eliminator Features: True Arcade Action Graphics(tm) and sound, "look ahead" radar function, real time, advancing play levels, hyperspace and disruptor functions.



SEA DRAGON: You're the captain of the U.S.S. Sea Dragon, a submarine of the Skate class. Your mission is to infiltrate a dangerous enemy-controlled channel. You must destroy rising underwater mines while avoiding wave after wave of additional dangers, including depth charges dropped by battleships, automatic lasers, and deadly enemy attack bases. You must successfully fight an unyielding enemy and an unknown undersea channel — and the only options are total victory or a salty grave.
Sea Dragon Features: Arcade Action Graphics(tm) and sound, 29 (!!!) screens of horizontally scrolling seascape, advancing skill levels and two player option. Early reviews are outstanding!

REAR GUARD: Waves of Cyborg ships are attempting to attack your fleet's crew pods. If they penetrate your defenses, they'll kamikaze your fleet. The enemy is relentless, your crew pods weaponless, your firepower endless. Are you made of the right stuff to successfully defend your fleet?! FIND OUT NOW!
Rear Guard Features: Incredible 3-D scrolling planetscape, sound, Arcade Action Graphics(tm), nearly two dozen types of enemy craft, two player option and running high score.

Prices subject to change without notice

Adventure INTERNATIONAL
a subsidiary of Scott Adams, Inc.

~19

To order, see your local dealer. If he does not have the program you want, then call 1-800-327-7172 (orders only) or write for our free catalog.

Published by ADVENTURE INTERNATIONAL
a subsidiary of Scott Adams, Inc.
BOX 3435 • LONGWOOD, FL 32750 • (305) 830-8194

Brainstorm

by Richard Ramella

If you like figuring out how to complete number sequences like those in IQ tests, you'll love Brainstorm. Just don't ask how it works.

It's 1955 and I'm sitting in a chalk-dusty, steam-heated classroom at Rockenroehl High School driving myself nuts over whether to choose A, B, C, D, or none of the above.

It's IQ test day.

Despite the frustrations, I loved IQ tests. I always hoped my many harebrained guesses might by some miracle all be correct, producing a flood of scholarship offers. It never happened.

The type of question that most often led to my wild guesses was the number sequence. I wrote the program Brainstorm somewhat as a game for my children, but also to check whether my shortcoming in number sequences was a genetic flaw. Brainstorm generates number sequences like the ones in IQ tests.

Although the program was written in Level II, it will work in Level I with two line changes, which I will give later.

Brainstorm is simply written, but don't let that lull you into a false sense of security. The problems range from the simple to some real brainbusters.

The classic number sequence of an IQ test is usually made up of a list of five numbers followed by four multiple-choice answers, one of which is correct. The elusive sixth number continues a logical pattern that can be understood by comparing the relationships of the first five numbers.

Brainstorm uses random number selections to branch to and set up 10 different logic schemes. Some of these 10 can go into subschemes. The result is that the variety of problems is limitless, though I think you'll fall by the wayside with brain fever or figure out all the patterns long before the number orders become repetitively boring.

At the start of the program, the computer randomly chooses one of the 10 logic schemes, branches to it, and prints a sequence of five numbers. It prompts, "Next number in series?" and waits for your answer. Key the answer and tap enter. If you don't know the answer, just tap enter. If you're right, the computer says so and goes on to the next problem. If you're wrong, the computer gives you the answer. Then it gives you a chance to study the six numbers at leisure before tapping enter to continue.

No score is kept. The program does not test your intelligence in terms of assigning you a neat IQ score. It's a game.

The program will work in Level I with the following two line changes:

```
1180 IF (L=5)*(A=1)
  THEN F=F+X: GOTO 1210
1190 IF (L=5)*(A=2)
  THEN F=F-X: GOTO 1210
```

Because I felt a bit devilish when I wrote Brainstorm, I haven't included any explanations of the logic schemes. One could figure them out from the program, but it's probably more trouble than it's worth.

If any number sequence stumps you, send me the sequence in a stamped, self-addressed envelope, and I'll reply with the answer and the logic involved. Please, send five sequences or fewer. I'm still no good at number sequences, but I did write the program and know how it works.

And if I can't figure it out, my 9-year-old daughter can. Early results indicate my mental deficiencies were not passed on to my offspring. ■

Richard Ramella is 80 Micro*'s Fun House columnist. He can be reached at 1493 Mountain View Ave., Chico, CA 95926.*

Program Listing

```
100 REM * BRAINSTORM: IQ NUMBER SEQUENCES *
110 CLS
120 A=RND(10)
130 ON A GOTO 140,210,320,420,530,640,740,830,960,1100
140 F=RND(10)
150 S=RND(10)
160 FOR L=1 TO 5
170 PRINT F;
180 F=F+S
190 IF L=5 GOTO 1210
200 NEXT L
210 G=RND(8)
220 S=RND(12)
230 IF S<=G GOTO 220
240 PRINT G;S;
250 FOR L=1 TO 4
260 F=G+S
```

Listing continues

The Key Box

Model I
4K RAM
Basic Level II (or Level I with two line changes)

Listing continued

```
270 IF L=4 GOTO 1210
280 PRINT F;
290 G=S
300 S=F
310 NEXT L
320 F=RND(20)
330 S=RND(10)
340 T=RND(10)
350 FOR L=1 TO 6
360 N=N+1
370 IF N=1 F=F+S
380 IF N=2 F=F+T: N=0
390 IF L=6 GOTO 1210
400 PRINT F;
410 NEXT L
420 F=RND(10)+5
430 S=RND(6)
440 T=RND(10)
450 IF T=<S GOTO 440
460 FOR L=1 TO 5
470 N=N+1
480 PRINT F;
490 IF N=1 F=F-S
500 IF N=2 F=F+T: N=0
510 IF L=5 GOTO 1210
520 NEXT L
530 S=RND(20)
540 F=RND(20)
550 T=RND(10)
560 FOR L=1 TO 6
570 N=N+1
580 IF N=1 S=S+T
590 IF N=2 F=F+T
600 IF L=6 GOTO 1210
610 IF N=1 PRINT S;
620 IF N=2 PRINT F;: N=0
630 NEXT L
640 T=RND(5)
650 S=RND(5)
660 PRINT T;S;
670 F=T+S
680 PRINT F;
690 FOR L=1 TO 3
700 F=F+F
710 IF L=3 GOTO 1210
720 PRINT F;
730 NEXT L
740 F=RND(4)
750 T=RND(4)+1
760 IF T=F GOTO 750
770 PRINT F;
780 FOR L=1 TO 5
790 F=F*T
800 IF L=5 GOTO 1210
810 PRINT F;
820 NEXT L
830 T=RND(5)
840 S=RND(8)+4
850 IF S=<T GOTO 1020
860 A=RND(2)
870 PRINT T;S;
880 FOR L=1 TO 4
890 IF A=1 F=T+S-1
900 IF A=2 F=T+S+1
910 IF L=4 GOTO 1210
920 PRINT F;
930 T=S
940 S=F
950 NEXT L
960 G=RND(3)
970 S=RND(7)
980 IF S<G GOTO 970
```

Listing continues

Listing continued

```
990 T=RND(9)
1000 IF T<S GOTO 990
1010 PRINT G;S;T;
1020 FOR L=1 TO 3
1030 F=G+S+T
1040 IF L=3 GOTO 1210
1050 PRINT F;
1060 G=S
1070 S=T
1080 T=F
1090 NEXT L
1100 F=RND(1000)+60
1110 IF F/32<>INT(F/32) GOTO 1100
1120 X=RND(3)
1130 A=RND(2)
1140 FOR L=1 TO 6
1150 IF A=1 PRINT F+X;
1160 IF A=2 PRINT F-X;
1170 F=F/2
1180 IF L=5 AND A=1 THEN F=F+X: GOTO 1210
1190 IF L=5 AND A=2 THEN F=F-X: GOTO 1210
1200 NEXT L
1210 PRINT
1220 PRINT
1230 INPUT "NEXT NUMBER IN SERIES";B
1240 PRINT
1250 IF B=F PRINT "RIGHT": GOTO 1290
1260 PRINT "NO, ANSWER IS";F
1270 PRINT
1280 INPUT "ENTER TO CONTINUE";X: GOTO 110
1290 FOR T=1 TO 500
1300 NEXT T
1310 GOTO 110
```


See List of Advertisers on Page 563

More Memory for Peanuts

by James Schaefer

Adding memory to your Model III is not as hard as it seems, so don't be intimidated. But, remember that you may void your warranty.

If you dream about installing more memory in your Model III computer but haven't done it yet because you think it's too difficult or expensive, just put aside your fears and follow this step-by-step guide.

First you need to buy some memory chips. These chips are sold in sets of eight for about $20 per set. Each set adds 16K to the computer. The Model III uses 4116 chips with a speed of 200 nanoseconds.

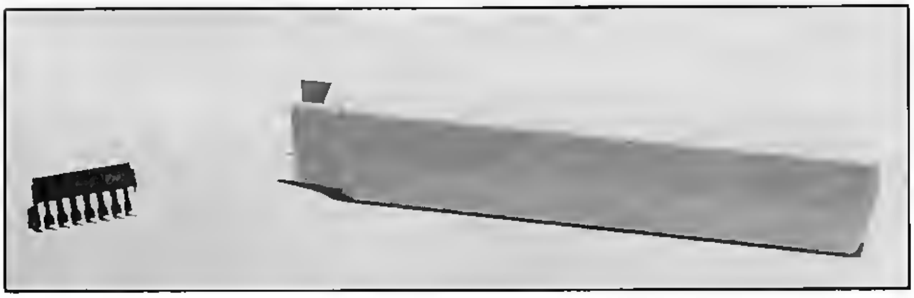


Photo 1

Photo 1 shows a single memory chip and an inexpensive tool ($5) that holds the chip. The chip has 16 connector pins, eight on each side. The insertion tool holds the pins straight as you plug in the chips. You can install the memory chips without the insertion tool, but the tool makes the job easier.

When you are ready to install the memory chips spread a soft cloth on a large table. This protects your computer case from scratches. Find the screw on the back of the case and remove it. Next turn the computer over and remove the screws from the bottom of the case. These are of different

The Key Box

Model III
16K RAM

Photo 2

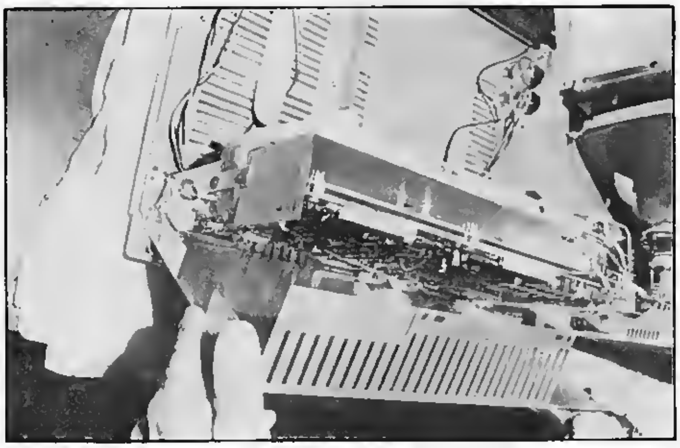


Photo 3

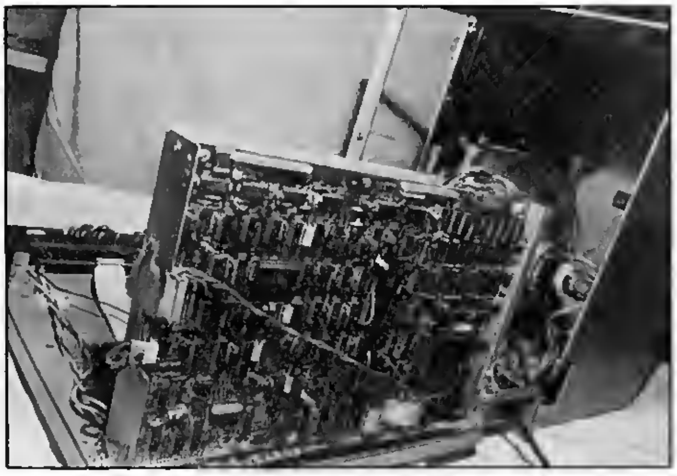


Photo 4

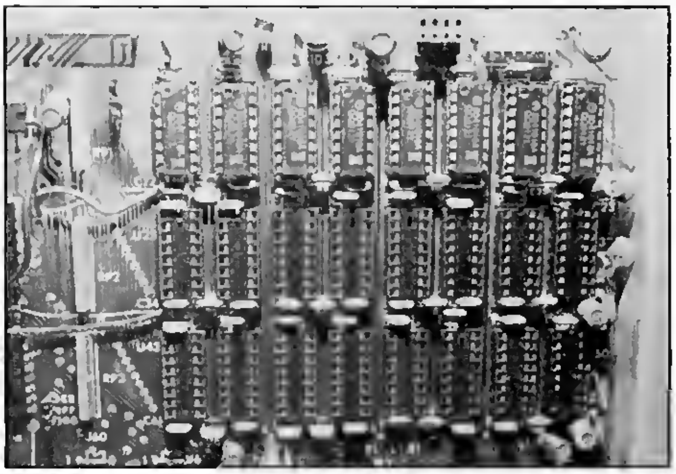


Photo 5

lengths, so remember where each one came from. Hold the top and bottom of the computer together and turn it right side up.

Look at the metal chassis at the rear of the bottom half of the computer in Photo 2. The small end of the picture tube fits into the narrow portion of the chassis on the right side. There is only about ½ inch of clearance between the picture tube and the chassis. You must carefully lift the top half of the computer straight up. After the picture tube has cleared the top of the chassis place the top half of the computer to the side as in the photo.

Remove the metal shield (Photo 3). This shield helps reduce EMI radiation levels which cause interference in your home radio and tv. Some of the first Model III computers did not have this shield installed.

Photo 4 shows the main circuit board.

Photo 5 shows an enlarged view of the top right corner of the circuit board. Here you see three rows of eight sockets each. The top row is filled with memory chips. If you have a 16K computer they are filled with 4116 chips. Now look at the top of each chip in the first row. There is a small dot at the top of each chip. These dots help tell you how to plug in the chip. If the chips you buy do not have a marker on one end then put the chips in so the writing goes in the same direction as the chips in the first row.

In Photo 6 I am using the insertion tool to put in a memory chip in the second row. If you want to add only 16K of memory to a 16K computer, then fill the second row with memory chips. Use the third row when you want to add the third set of 16K memory chips to bring the RAM up to 48K.

After each chip is inserted make sure it is fully seated in the socket by pressing it with your thumb (Photo 7). Use a little caution! You want to seat the chip, not crack the main circuit board. Also, make sure that all pins on the memory chip have gone into the socket holes. If one pin bends under the chip pull it out and try again.

When you have installed all the memory, plug in the computer. Turn it on and test the computer with ? MEM. It will take a little longer for the memory size to come back because there is more memory for the computer to look at. In a 48K Model III you should get 48082 and in a 32K machine you should get 32768. If everything looks correct turn off the computer and unplug it.

You are now ready to put the computer back together. Start by replacing the metal shield. Make sure the braided ground strap lug goes under one of the screws at the top of the narrow portion of the chassis. Carefully lift the top half of the computer over the bottom half and gently lower it into place. Remember the back of the picture tube has a clearance of only ½ inch. Hold both halves together and turn the computer upside-down. Replace the screws that hold the two halves together. Turn the computer right side up. Then resecure the screw at the back of the case. ■

James Schaefer (33 Jackson Road, Berlin, NJ 08009) is a computer consultant for two schools.

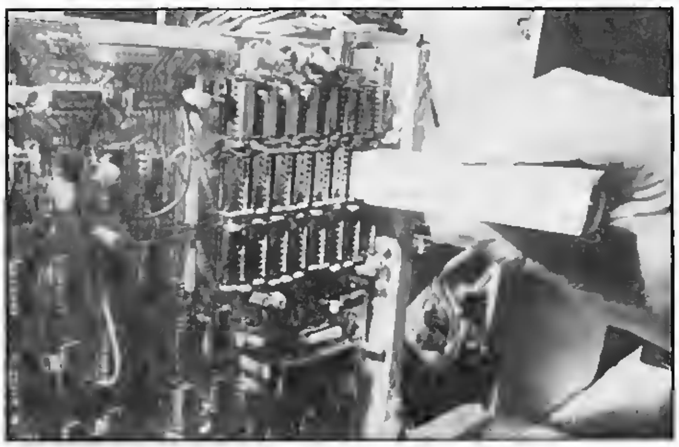


Photo 6

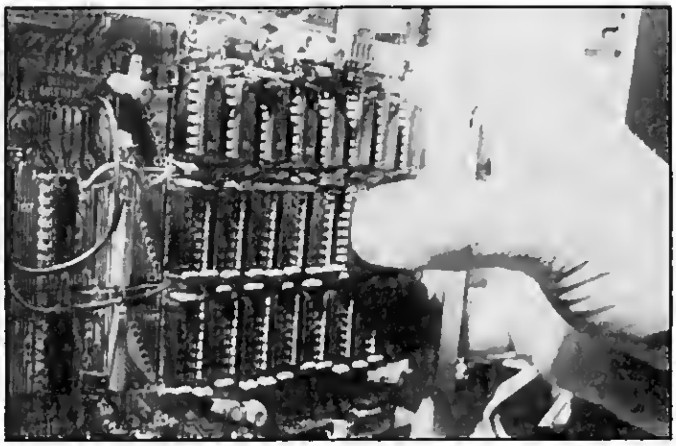


Photo 7

COMPUTEX

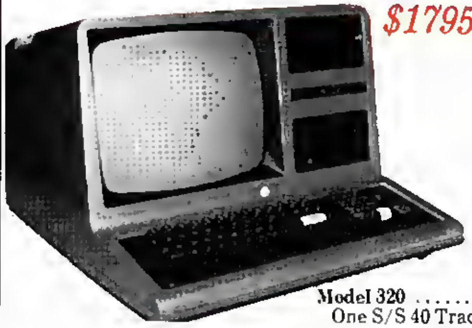
a subsidiary of Worldwide Data Systems Inc.

SPECIAL

~27

Dual Drive Model III

CUSTOM BUILT
*M321
$1795.00



Model 320 .... $1695.00
One S/S 40 Track Drive

*Model 321 .... ~~$1895.00~~
Two S/S 40 Track Drive

Model 322 .... $2295.00
Two D/S 40 Track Drives

Model 324 .... $2595.00
Two D/S 80 Track Drives

Model 325 .... $3995.00
One TM602 Winchester &
One S/S 40 Track Drive

Model 326 .... $4495.00
One TM603 Winchester &
One D/S 80 Track Drive

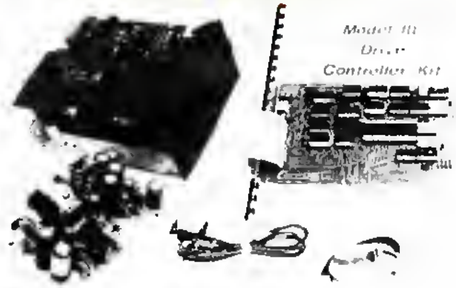
FEATURES

DOS PLUS 3.4 Operating System
Winchester Expansion Option
Gold Plated Edge Cards
Fully Socketed Drive Controller
Switching Power Supply
System Dust Cover
Expansion Capabilities
90 Day Warranty
or
1 Year Warranty Available

P-31 Green Phosphor CRT $89.00

All customized Model III systems from Computex now include at *no extra charge* DOSPLUS 3.4. A Computex customized system starts out as a basic Model III, 32K RAM is added to increase the storage to 48K. Add to this our Model III drive controller with Winchester expansion option and the appropriate Tandon or MPI disk drives per your requirements. The system is then tested vigorously for 48 hours to assure you of peak performance when the system arrives at your doorstep.

COMPLETE MODEL III DRIVE KITS



Complete Model III Disk Drive Kits include the Computex drive controller *with capabilities of Winchester drive expansion.* Virtually everything you will need to install disk drives in your Model III is included - We even include one of the most popular DOS systems available, DOS PLUS 3.4.

M3DK0 Drive Kit with 1 Single Sided 40 Track Drive
$495.00

M3DK1 Drive Kit with 2 Single Sided 40 Track Drives
$699.00

M3DK2 Drive Kit with 2 Double sided 40 Track Drives
$995.00

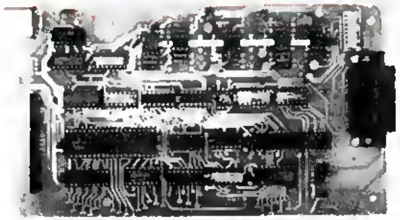
M3DK4 Drive Kit with 2 Double Sided 80 Track Drives
$1150.00

Model III Disk Drive Kit Less Drive's & DOS

SPECIAL $249.95

MODEL III INTERNAL COMMUNICATIONS BOARD

NEW !



The newest addition to the Computex line of Model III add on kits is our **M3CB1** *Communications board complete with* **RS232** *and* **DIRECT CONNECT MODEM.** *The kit can be supplied in either total kit form or as a partially assembled unit. The partially assembled unit only requires installation of the integrated circuits, all other components have been installed and tested. The* **M3CB1** *mounts in the same location as the Radio Shack RS232 board and installs easily in less than one hour.*

FEATURES

•300 to 19,200 Baud RS232•
•300 Baud Full Duplex Operation•
•Uses FSK Modulation Techniques•
•Supports Originate or Answer Modes•
•Will Operate DTE or DCE Equipment•

M3CB1-1	P.C.B. & Manual	$34.95
M3CB1-2	Assembled & Tested RS232	$69.95
M3CB1-3	Complete Kit	$129.95
M3CB1-4	Partially Assembled Kit	$169.95

Our qualified technicians are available for technical questions at [713] 488-8022 every day from 4:00 to 5:00 PM CST

MPI & Tandon 5.25" DISK DRIVES

•Single or Double Density Operation•
•3 - 5 ms Track to Track Access Time•
•48 Hr. Burn in with Pre/Post Testing•
•Compatible with TRS80®, Zenith, IBM•

All disk drives from Computex are fully tested for Radial Head Alignment, Track 0 Switch, Speed, Instantaneous Speed Variation ISV, Azimuth, then verified for data storage integrity, reliability and systems compatibility.

MPI	Tandon	Description	Price
B51	TM100-1	Single Sided 40 Track	$219.95
B52	TM100-2	Double Sided 40 Track	$299.95
B91	TM100-3	Single Sided 80 Track	$299.95
B92	TM100-4	Double Sided 80 Track	$399.95

Power Supply and Case $49.95

SPECIAL $249.95

40 Track Single Sided Disk Drive
Power Supply & Case

INTRODUCING THE

PROMETHEUS 5

6.3M Byte External $1395.00

WINCHESTER FOR THE MODEL III *Complete*

That's Right! Now you can have over 5M Bytes of formatted data storage (Options are available to increase from 5 to over 30 M Bytes) by simply plugging the Prometheus 5 into the I/O connector on your Model III.

Developed by Computex in conjunction with leading engineers in the Aerospace Industry, the Prometheus 5 offers quality and reliability using the latest advancements in technology. For example; The Winchester drive used in the system is from Tandon Magnetics, (the leading manufacturer of disk drives). The Winchester controller provides CRC generation / checking and error code correction circuitry, the fully regulated power supply has current and overvoltage protection as well as EMI and transient filtering on the input line.

The Prometheus 5 is operated by DOS PLUS 4.0, the leading Winchester Operating System from Micro Systems Software, and provides capabilities not yet thought of in most other DOS systems.

So--If you're considering mass storage for your Model III, Consider the Prometheus 5 for quality, reliability, and excellence in mass storage peripherals.

Call or Write for brochure and picture describing the all new

PROMETHEUS 5

AVAILABLE EXCLUSIVELY FROM COMPUTEX

SUPER SAVERS FROM COMPUTEX

PRINTERS

C Itoh 8510 Prowriter I Parallel	*$549.00*
C Itoh 8510 Prowriter I Serial	*695.00*
C Itoh 1550 Prowriter II Parallel	*750.00*
C Itoh 1550 Prowriter II Serial	*850.00*
C Itoh F10-40cps Parallel	*1550.00*
C Itoh F10-40cps Serial	*1650.00*
C Itoh F10-40cps IEEE488	*1650.00*
Smith Corona TP-1	*650.00*

MODEMS 300 BAUD

Novation CAT Acoustic 300 Baud Ans/Org	*$140.00*
Novation DCAT Ans/Org D.C.	*175.00*
Novation Auto CAT Auto Ans/Org D.C.	*215.00*
UDS103 OALP Ans/Org D.C.	*175.00*
UDS103 JLP Auto Ans/Org D.C.	*209.00*
Lexicon LEX11 Acoustic Ans/Org	*135.00*

FURNITURE by DATADECKS

All furniture features mar and scratch resistant laminated tops, fully adjustable heights, easy assembly, moulded edges, with finishes available in either Oak or Putty.

Call or write for a free full color brochure that truly shows the beauty of the Data Deck line of computer furniture.

Universal Printer Stand 27 * 16	*$115.00*
36 * 30 Basic Work Station	*155.00*
48 * 30 Basic Work Station	*178.00*
60 * 30 Basic Work Station	*195.00*
Upper Shelf Assembly for 36 * 30 Desk	*85.00*
Upper Shelf Assembly for 48 * 30 Desk	*95.00*
Upper Shelf Assembly for 60 * 30 Desk	*109.00*
Disk Drive Module for 48 or 60 Desk	*66.00*
Lower Shelf Assembly for 36 Desk	*66.00*

PLOTTERS

Watanabe WX4671	*$1195.00*

MEDIA * ACCESSORIES * BOOKS

Verbatim 525-01 S.S. D.D.	*$26.95*
Plastic Library Box 5†	*2.60*
TRS80 Disk & Other Mysteries	*19.95*
Microsoft Basic Decoded	*24.95*
The Custom TRS80 & Other Mysteries	*24.95*
Basic Faster & Better	*24.95*
Model I or Model III Joysticks	*39.95*

SOFTWARE

DOS PLUS 3.4	*$119.00*
Starfighter A.I.	*24.95*
The Eliminator A.I.	*21.95*
Z Chess A.I.	*24.95*
Stones of Sisyphus A.I.	*24.95*
Escape from Traam A.I.	*17.95*
Planetoids A.I.	*17.95*
Sky Warrior A.I.	*17.95*
Missile Attack A.I.	*17.95*
Space Intruders A.I.	*17.95*
Showdown A.I.	*17.95*
Combat A.I.	*17.95*
Amored Patrol A.I.	*21.95*
Maxi Manager A.I.	*129.95*
Maxi Cras A.I.	*89.95*

CABLES

C Itoh - Model III Printer	*29.95*
Drive Extender	*12.95*
2 Drive Cable	*29.95*
4 Drive Cable	*39.95*

INTERFACE EXPANSION KITS FOR TRS80 COMPUTERS

The MDX Series Interface Expansion Boards provide a wide range of features for the TRS80® Computers. Bare boards or complete kits are available for the MDX-2 (Model I Expansion I/F and the MDX-3 Model III Expansion I/F) which will allow you to upgrade your system by assembling only those sections or features that you need when you need them. P.C. Boards are silkscreened and soldermasked for ease in assembly and the assembly / user's manual supplied with each kit includes theoretical operation as well as step by step assembly and installation instructions.

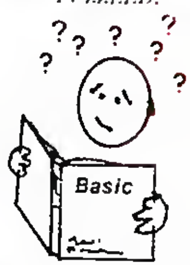
MDX-2 Model I Expansion I/F Kit	**$269.95**
RS232 Serial I/F Kit	24.95
RAM Expansion Kit (less RAM)	17.95
Disk Controller Kit	31.95
Hardware & Socket Kit	29.95
Power Supply Kit	29.95
MDX-3 Model III Expansion I/F	**$259.95**
Disk Controller Kit	89.95
RS232 Serial I/F Kit	79.95
Phone Modem Kit	39.95

Additional Kits Available - Call For Quotes

If you're considering a IBM, Apple III, or Model 16 computer for your business needs, consider the NorthStar ADVANTAGE - We think you'll be surprised! Why? Simple. The NorthStar ADVANTAGE offers far more capability than any of the aforementioned systems. For example; The NorthStar ADVANTAGE features high resolution graphics, Full CP/M support, 64K RAM (upgradeable), 720K Bytes disk storage, tried and proven software, and the capability of growing with your needs. Optional interfaces are available that will allow you to run 8 or 16 bit software or upgrade to a multiuser system by simple adding NorthNet. Perhaps the most incredible part is the price - $3595.00 (less than you'll pay to upgrade a Model III to the same capabilities).

So if your tired of playing games, or you need a serious system for your business, contact Fred Haskin at (713)488-4215 for further information and a demonstration of the NorthStar ADVANTAGE

PUZZLED?



SEE OUR AD ON PAGE 430

COMPUTEX

a subsidiary of World Wide Data Systems Inc.

17321 El Dorado Blvd.
Webster, Texas 77598

(713)488-8022

~27

CALL OR WRITE FOR NEW CATALOG

Here's How To Order

Phone orders welcomed with VISA, MC, or American Express. Net 30 Terms available to well rated D&B firms, Universities, and Government Organizations. Sorry, COD orders are limited to $100.00. Prices quoted do not include shipping, are available mail order only, and are valid during the month of publication. Prices are subject to change wihtout notice. TRS80, Model I, & Model III are registered trademarks of Radio Shack a Tandy Corporation.

Fast Tape Operating System

by Michael Pollard

Fast Tape Operating System, FTOS, is an Assembly-language program that replaces the tape operating system in ROM with one in RAM that is faster and more powerful.

Here's what FTOS can do:

- Load and save an entire array with one command.
- Input and output at 1,800 baud.
- Save Assembly-language routines.
- CLOAD Basic and Assembly-language programs.
- Find the beginning of the program or array on the tape, so you don't have to position the tape on a blank spot to load.
- Display the six-character file name of a program or array.
- Load and run Assembly-language programs with a single command.
- Allow the break key to stop the cassette during a load.

FTOS completes all these tasks with less than 1K of memory. It resides at the top of 16K, or anywhere you wish to relocate it, and uses the disk commands, making it incompatible with a disk system.

FTOS lets you use disk commands to increase the speed of storage time in your cassette system.

The Disk Commands

To understand how FTOS uses the disk commands, type save and hit enter. You get an L3 error. Now type POKE 16800,195: POKE 16801,0: POKE 16802,0 and hit enter. Type in "save" and enter again and you get the memory-size prompt.

Save is a disk command and if you use it without a disk, you normally get an L3 error. When the Basic interpreter in ROM sees the instruction Save, it calls (like a GOSUB in Basic) the address 16800. There is usually a jump to the L3 error routine there, but those three POKEs put a JP 00 (GOTO 00), which is where the power-up routine starts, and you get the memory-size prompt instead of an L3 error.

FTOS has patches at Save and at the other disk commands it uses. These patches don't jump to 00 but to specific routines in FTOS. That is how FTOS is hooked up.

Fast Input/Output

Using ROM I/O, programs and data are sent to and from the cassette, one byte at a time. Each byte is composed of 8 clock bits and 8 data bits. A pulse between clock bits represents 1, and a 0 is represented by lack of a pulse. To increase I/O speed FTOS outputs (and inputs) 1 clock bit followed by the 8 bits of the byte. This means that there are only 9 bits (1 clock, 8 data) per byte instead of 16 bits (8 clock, 8 data). This creates faster code and decreases loading errors, since the output is about half its usual length.

To further increase speed, the spaces between pulses as well as the pulses themselves, are shortened. As a result of these modifications, the speed of I/O will be about 3 1/2 times faster than that of normal cassette I/O.

Added Reliability

To increase reliability, the tape formats of ROM have been replaced by one format (see Fig. 1). This format is not only more reliable, but also more flexible. It is similar to ROM's tape format, but it does have two major differences. One is the synchronization code. ROM uses a one-byte synchronization code to point to the beginning of valid data. Unless the tape head is positioned over a blank spot, ROM will probably find a false sync code, leading to a bad load. To solve this problem, FTOS has a two-byte sync code. The tape can be positioned anywhere, even in the middle of data, and FTOS still finds the beginning of the next program. The block header code is also two bytes long to help recover data in the event of a bad read.

The other major difference is the addition of a type code. A 0 code is for Assembly language, a 1 code is for Basic, and a 2 code is for arrays. This tells FTOS which data type is being loaded. The type code is necessary since FTOS uses only one tape format.

If you get a checksum error (CE next to the asterisks) while loading, but loading still completes, probably only a small part of the program is incorrect. If the asterisks disappear, however, the program being read has an irrecoverable error. FTOS is looking for another program, so rewind the tape, adjust the volume, and try again.

Array I/O Made Easy

The most powerful statements of FTOS are the array commands: Open and Close. These commands input and output whole arrays to and from the cassette recorder. Figure 2 shows two

The Key Box
Model I
16K RAM
Assembly Language
Cassette Basic

programs for outputting a 100-element array, one with FTOS and one without. The listings show how much easier it is to program, since the formatting is an automatic function. The time difference between the two shows how much faster FTOS is than the ROM system. If you wish to output only part of an array, it can be put into a smaller array and then output to tape.

Tape Directory

FTOS can read anything it puts on tape. If you have ever mixed System, Basic, and PRINT# data, as well as EDTASM source code on the same tape, you know that separating them is almost impossible. If a name is specified when loading an FTOS tape, all the programs and arrays read before the specified tape will have their names printed. This allows you to see what is stored on a cassette.

Commands

Save name: Position tape and put recorder in record mode. Type save and a name of up to six characters. Hit enter and the resident Basic program will be stored on tape. If no name is specified, the name field will contain blanks.

Put name: Position tape and put recorder in record mode. Type put and a name of up to six characters. Press enter, and the program asks you to enter the four-digit hexadecimal starting address of the program. After the fourth digit is entered, enter the ending address. Then enter the entry-point address, and the recorder will start recording.

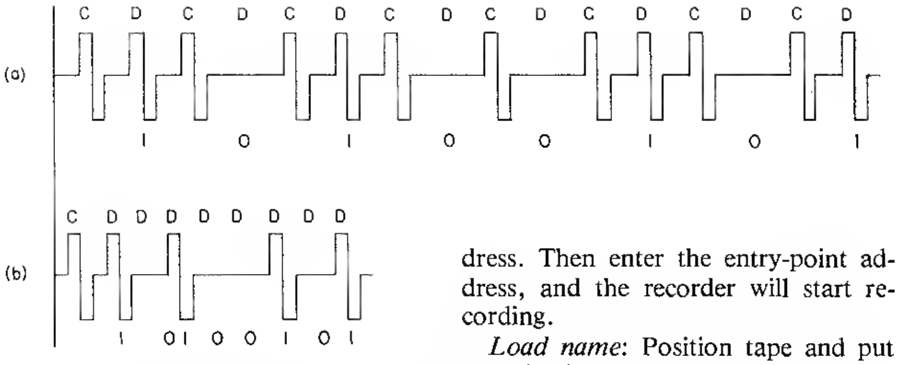


Fig. 1. Comparison between the ROM format (a) for A5 hexadecimal and the FTBS formab (b) for A5H (C = clock, D = data)

256 bytes—zeros
2 bytes—sync code (A5,A6)
6 bytes—name field
1 byte—type code (0,1,2)
2 bytes—entry point

---

1 or more data blocks

---

1 byte—zero
2 bytes—block header code (3C,3D)
1 byte—block size (0 if 256 bytes)
2 bytes—beginning memory location
1–256 bytes—data
1 byte—checksum

---

1 byte—end code

Fig. 2. FTOS Tape Format

Load name: Position tape and put recorder in play mode. The tape need not be positioned on a blank spot to find the beginning of the program. If no name is specified, the first program read is loaded. If a name is specified, FTOS displays the names of all the programs it reads until the specified program is loaded. If the program loaded is an Assembly-language program, you are prompted to enter an entry-point address in hexadecimal. You can enter a starting-point address, or hit the break key, which brings you back to Ready, or you may hit enter, which jumps to the entry point specified on the tape.

Get name: This is the same as Load. However, if an Assembly-language program is loaded, control automatically jumps to the program's entry point.

CMD name: This command functions the same as CLOAD, but it also checks Assembly-language programs.

Close name: This saves an entire array on tape. The first two characters in the name must match those of an existing array in memory.

Open name: This loads an entire array. The data loaded in must go into an array of the same type and size as that from which it was saved. Since Basic only recognizes the first two characters, different sets of data may be used in a single array by having two different names with the same first two characters. ■

Michael Pollard can be reached at 4407 W. Walnut, Soquel, CA 95073.

Program Listing

```
                  ;       (C)     "FTOS" - A CASSETTE ROUTINE
                  ;                 BY M. RODNEY POLLARD
                  ;                    MAY 1982
3C40                      ORG     3C40H
3C40 46                   DEFM    'FTOS - 7CA3H-7FFFH  '
400F                      ORG     400FH           ;RST 30H
400F C3037D               JP      CP1
4173                      ORG     4173H           ;CMD
4173 C3F37C               JP      CMD
417F                      ORG     417FH           ;GET
417F C3D77E               JP      GET
4182                      ORG     4182H           ;PUT
4182 C3887D               JP      PUT
4188                      ORG     4188H           ;LOAD
4188 C3DB7E               JP      BR
41A0                      ORG     41A0H           ;SAVE
41A0 C3087E               JP      BW
4185                      ORG     4185H           ;CLOSE
4185 C3C97D               JP      PR
4179                      ORG     4179H           ;OPEN
4179 C3D57D               JP      INP
7CA3                      ORG     7CA3H           ;BEGINNING OF FTOS
7CA3 AF           CKA     XOR     A
7CA4 32D77F               LD      (NAME+1),A
7CA7 E5           CK      PUSH    HL              ;LOAD NAME
7CA8 D9                   EXX
7CA9 E1                   POP     HL
7CAA 2B                   DEC     HL
7CAB CDC77E               CALL    BLANK           ;BLANK NAME FIELD
7CAE 0606                 LD      B,6
7CB0 11D67F               LD      DE,NAME
7CB3 D7           CK1     RST     10H             ;CHECKS CHARACTERS
7CB4 2809                 JR      Z,CK2           ;EXIT
7CB6 FE2F                 CP      2FH
7CB8 38F9                 JR      C,CK1
7CBA 12                   LD      (DE),A
7CBB 13                   INC     DE
7CBC 04                   INC     B
```

Listing continues

```
10 DIM AC(100)
20 FOR J = 1 TO 10
30 FOR I = 1 TO 10:PRINT# - 1,AC(I),
AC(I + 1),AC(I + 2),AC(I + 3),AC(I + 4),
AC(I + 5),AC(I + 6),AC(I + 7),AC(I + 8),
AC(I + 9):NEXT I
40 NEXT J

  (TIME = 48 SECONDS)

10 DIM AC(100)
20 CLOSE AC

  (TIME = 4 SECONDS)
```

Figure 3

Listing continued

```
7CBD 18F4      00410        JR      CK1
7CBF E5        00420 CK2    PUSH    HL
7CC0 D9        00430        EXX
7CC1 E1        00440        POP     HL
7CC2 D9        00450        EXX
7CC3 C9        00460        RET
7CC4 B7        00470 TS     OR      A               ;TABLE SEARCH
7CC5 2AFB40    00480        LD      HL,(40FBH)      ;TOP OF ARRAY TABLE
7CC8 ED4BFD40  00490        LD      BC,(40FDH)      ;BOTTOM OF ARRAY TABLE+1
7CCC 23        00540 TS1    INC     HL
7CCD ED5BD67F  00550        LD      DE,(NAME)
7CD1 7E        00560        LD      A,(HL)
7CD2 BA        00570        CP      D
7CD3 2006      00580        JR      NZ,TS2
7CD5 23        00590        INC     HL
7CD6 7E        00600        LD      A,(HL)
7CD7 BB        00610        CP      E
7CD8 2813      00620        JR      Z,TS3
7CDA 2B        00630        DEC     HL
7CDB 23        00640 TS2    INC     HL
7CDC 23        00650        INC     HL
7CDD 5E        00660        LD      E,(HL)
7CDE 23        00670        INC     HL
7CDF 56        00680        LD      D,(HL)
7CE0 23        00690        INC     HL
7CE1 19        00700        ADD     HL,DE
7CE2 E5        00710        PUSH    HL
7CE3 B7        00720        OR      A
7CE4 ED42      00730        SBC     HL,BC
7CE6 E1        00740        POP     HL
7CE7 FACC7C    00750        JP      M,TS1
7CEA E1        00760        POP     HL
7CEB D9        00770        EXX
7CEC C9        00780        RET
7CED 23        00790 TS3    INC     HL
7CEE 5E        00800        LD      E,(HL)
7CEF 23        00810        INC     HL
7CF0 56        00820        LD      D,(HL)
7CF1 23        00830        INC     HL
7CF2 C9        00840        RET                 ;HL HAS START, DE HAS LENGTH
7CF3 F5        00850 CMD    PUSH    AF
7CF4 3EF7      00860        LD      A,0F7H          ;RST 30H
7CF6 32487F    00870        LD      (BR7),A
7CF9 CDDB7E    00880        CALL    BR
7CFC 3E77      00890        LD      A,77H           ;FIX IT
7CFE 32487F    00900        LD      (BR7),A
7D01 F1        00910        POP     AF
7D02 C9        00920        RET
7D03 BE        00930 CP1    CP      (HL)
7D04 C8        00940        RET     Z
7D05 21FB7F    00950        LD      HL,BAD
7D08 3E0D      00960        LD      A,0DH
7D0A CD3300    00970        CALL    33H             ;DISPLAY CONTENTS OF A
7D0D CDA728    00980        CALL    28A7H
7D10 E1        00990        POP     HL
7D11 C3727F    01000        JP      BR5
7D14 AF        01010 LEAD   XOR     A
7D15 CD1202    01020        CALL    212H
7D18 0600      01030        LD      B,0
7D1A AF        01040        XOR     A
7D1B CD8A7E    01050 Z11    CALL    WRITE
7D1E 10FB      01060        DJNZ    Z11
7D20 C9        01070        RET
7D21 0604      01080 SYNC   LD      B,4
7D23 3E20      01090        LD      A,20H
7D25 113C3C    01100        LD      DE,3C3CH
7D28 12        01110 SY1    LD      (DE),A
7D29 13        01120        INC     DE
7D2A 10FC      01130        DJNZ    SY1             ;BLANK ASTERISK AREA
7D2C AF        01140        XOR     A
7D2D CD1202    01150        CALL    212H            ;TURN ON CASSETTE
7D30 CDA67F    01160 K1     CALL    READ
7D33 FEA5      01170 L1     CP      0A5H            ;FIRST BYTE OF SYNC CODE
7D35 20F9      01180        JR      NZ,K1
7D37 CDA67F    01190 M1     CALL    READ
7D3A FEA6      01200        CP      0A6H            ;SECOND BYTE OF SYNC CODE
7D3C C8        01210        RET     Z
7D3D 18F4      01220        JR      L1
7D3F CD487D    01230 HEX    CALL    IN8
7D42 60        01240        LD      H,B
7D43 CD487D    01250        CALL    IN8
7D46 68        01260        LD      L,B
7D47 C9        01270        RET
7D48 CD5E7D    01280 IN8    CALL    INCH
7D4B 07        01290        RLCA
7D4C 07        01300        RLCA
7D4D 07        01310        RLCA
7D4E 07        01320        RLCA
7D4F 47        01330        LD      B,A
7D50 79        01340        LD      A,C
7D51 CD3A03    01350        CALL    33AH
7D54 CD5E7D    01360        CALL    INCH
7D57 80        01370        ADD     A,B
7D58 47        01380        LD      B,A
7D59 79        01390        LD      A,C
7D5A CD3A03    01400        CALL    33AH
7D5D C9        01410        RET
7D5E CD8403    01420 INCH   CALL    384H
7D61 FE01      01430        CP      1
7D63 281D      01440        JR      Z,IN2
7D65 FE0D      01450        CP      0DH
7D67 2006      01460        JR      NZ,IN4
7D69 2ADD7F    01470        LD      HL,(ENTRY)
7D6C F1        01480        POP     AF
7D6D F1        01490        POP     AF
7D6E C9        01500        RET
7D6F 4F        01510 IN4    LD      C,A
7D70 D630      01520        SUB     30H
7D72 FA5E7D    01530        JP      M,INCH
7D75 FE0A      01540        CP      0AH
7D77 F8        01550        RET     M
7D78 D607      01560        SUB     7
7D7A FA5E7D    01570        JP      M,INCH
7D7D FE10      01580        CP      10H
7D7F F8        01590        RET     M
7D80 18DC      01600        JR      INCH
7D82 E1        01610 IN2    POP     HL
7D83 E1        01620        POP     HL
7D84 E1        01630 IN3    POP     HL
7D85 C35A7E    01640        JP      BWA
7D88 CDA77C    01660 PUT    CALL    CK
7D8B 3E0D      01670        LD      A,0DH
7D8D CD3300    01680        CALL    33H
7D90 21E67F    01690        LD      HL,ST
7D93 CDA728    01700        CALL    28A7H           ;DISPLAY
7D96 CD3F7D    01710        CALL    HEX             ;CONVERT ASCII TO HEX
7D99 22E17F    01720        LD      (START),HL
7D9C 21ED7F    01730        LD      HL,ED
7D9F CDA728    01740        CALL    28A7H
7DA2 CD3F7D    01750        CALL    HEX
7DA5 22E37F    01760        LD      (END),HL
7DA8 CDBA7D    01770        CALL    EN
7DAB 3E0D      01780        LD      A,0DH
7DAD CD3300    01790        CALL    33H
7DB0 2AE37F    01800        LD      HL,(END)
7DB3 ED5BE17F  01810        LD      DE,(START)
7DB7 AF        01820        XOR     A               ;0 FOR ASSEMBLY LANGUAGE
7DB8 1860      01840        JR      BWD
7DBA 21F37F    01850 EN     LD      HL,ENT
7DBD CDA728    01860        CALL    28A7H
7DC0 D1        01870        POP     DE
```

Listing continues

Listing continued

```
7DC1 CD3F7D   Ø188Ø        CALL   HEX
7DC4 D5       Ø189Ø        PUSH   DE
7DC5 22DD7F   Ø19ØØ        LD     (ENTRY),HL
7DC8 C9       Ø191Ø        RET
7DC9 CDA37C   Ø192Ø PR     CALL   CKA
7DCC CDC47C   Ø193Ø        CALL   TS
7DCF EB       Ø196Ø        EX     DE,HL
7DDØ 19       Ø197Ø        ADD    HL,DE
7DD1 3EØ2     Ø198Ø        LD     A,2            ;TYPE CODE FOR ARRAYS
7DD3 1845     Ø199Ø        JR     BWD
7DD5 CDA37C   Ø2ØØØ INP    CALL   CKA
7DD8 CDC47C   Ø2Ø1Ø        CALL   TS
7DDB 11F17D   Ø2Ø2Ø        LD     DE,INP1
7DDE ED53437F Ø2Ø3Ø        LD     (BRP+1),DE
7DE2 E5       Ø2Ø4Ø        PUSH   HL
7DE3 DDE1     Ø2Ø5Ø        POP    IX
7DE5 11F77D   Ø2Ø6Ø        LD     DE,INP2
7DE8 ED537C7F Ø2Ø7Ø        LD     (BRJ+1),DE
7DEC Ø6ØØ     Ø2Ø8Ø        LD     B,Ø
7DEE C3DE7E   Ø2Ø9Ø        JP     BRI
7DF1 DDE5     Ø21ØØ INP1   PUSH   IX
7DF3 DD24     Ø212Ø        DEFW   24DDH          ;INC HX
7DF5 E1       Ø216Ø        POP    HL
7DF6 C9       Ø217Ø        RET
7DF7 11737F   Ø218Ø INP2   LD     DE,BRR
7DFA ED53437F Ø219Ø        LD     (BRP+1),DE
7DFE 11BA7D   Ø22ØØ        LD     DE,EN
7EØ1 ED537C7F Ø221Ø        LD     (BRJ+1),DE
7EØ5 E1       Ø222Ø        POP    HL
7EØ6 1852     Ø223Ø        JR     BWA
7EØ8 CDA77C   Ø225Ø BW     CALL   CK
7EØB 21191A   Ø226Ø BW3    LD     HL,1A19H       ;RETURN TO BASIC
7EØE 22DD7F   Ø227Ø        LD     (ENTRY),HL
7E11 ED5BA44Ø Ø228Ø        LD     DE,(4ØA4H)     ;BEGINNING ADDR OF BASIC
7E15 2AF94Ø   Ø229Ø        LD     HL,(4ØF9H)     ;ENDING ADDRESS OF BASIC
7E18 3EØ1     Ø2295        LD     A,1            ;TYPE CODE FOR BASIC
7E1A ED53E17F Ø23ØØ BWD    LD     (START),DE
7E1E 22E37F   Ø231Ø        LD     (END),HL
7E21 32DC7F   Ø233Ø        LD     (TYPE),A
7E24 B7       Ø234Ø        OR     A
7E25 ED52     Ø235Ø        SBC    HL,DE
7E27 23       Ø236Ø        INC    HL
7E28 22DF7F   Ø237Ø        LD     (COUNT),HL
7E2B 21D47F   Ø238Ø        LD     HL,HEAD
7E2E CD147D   Ø239Ø        CALL   LEAD           ;WRITE LEADER
7E31 Ø6ØB     Ø24ØØ        LD     B,ØBH
7E33 7E       Ø241Ø BW4    LD     A,(HL)
7E34 23       Ø242Ø        INC    HL
7E35 CD8A7E   Ø243Ø        CALL   WRITE
7E38 1ØF9     Ø244Ø        DJNZ   BW4
7E3A 2AE17F   Ø245Ø        LD     HL,(START)
7E3D 11EØ7F   Ø246Ø        LD     DE,COUNT+1
7E4Ø 1A       Ø247Ø BW5    LD     A,(DE)
7E41 B7       Ø248Ø        OR     A
7E42 28Ø8     Ø249Ø        JR     Z,BW6
7E44 3D       Ø25ØØ        DEC    A
7E45 12       Ø251Ø        LD     (DE),A
7E46 AF       Ø252Ø        XOR    A
7E47 CD5F7E   Ø253Ø        CALL   BW8
7E4A 18F4     Ø254Ø        JR     BW5
7E4C 3ADF7F   Ø255Ø BW6    LD     A,(COUNT)
7E4F B7       Ø256Ø        OR     A
7E5Ø 28Ø3     Ø257Ø        JR     Z,BW7
7E52 CD5F7E   Ø258Ø        CALL   BW8
7E55 3E23     Ø259Ø BW7    LD     A,23H          ;END CODE
7E57 CD8A7E   Ø26ØØ        CALL   WRITE
7E5A CDF8Ø1   Ø261Ø BWA    CALL   1F8H           ;TURN OFF CASSETTE
7E5D D9       Ø2614        EXX
7E5E C9       Ø262Ø        RET                   ;GO BACK TO BASIC
7E5F 47       Ø263Ø BW8    LD     B,A
7E6Ø 3EØØ     Ø2632        LD     A,Ø
7E62 CD8A7E   Ø2634        CALL   WRITE
7E65 3E3C     Ø264Ø        LD     A,3CH          ;BLOCK HEADER CODE
7E67 CD8A7E   Ø265Ø        CALL   WRITE
7E6A 3E3D     Ø2652        LD     A,3DH          ;BLOCK HEADER CODE
7E6C CD8A7E   Ø2654        CALL   WRITE
7E6F 78       Ø266Ø        LD     A,B
7E7Ø CD8A7E   Ø267Ø        CALL   WRITE
7E73 7D       Ø268Ø        LD     A,L
7E74 CD8A7E   Ø269Ø        CALL   WRITE
7E77 7C       Ø27ØØ        LD     A,H
7E78 CD8A7E   Ø271Ø        CALL   WRITE
7E7B ØEØØ     Ø272Ø        LD     C,Ø
7E7D 7E       Ø273Ø BW9    LD     A,(HL)
7E7E CD8A7E   Ø274Ø        CALL   WRITE
7E81 81       Ø275Ø        ADD    A,C
7E82 4F       Ø276Ø        LD     C,A
7E83 23       Ø277Ø        INC    HL
7E84 1ØF7     Ø278Ø        DJNZ   BW9
7E86 CD8A7E   Ø279Ø        CALL   WRITE
7E89 C9       Ø28ØØ        RET
7E8A E5       Ø281Ø WRITE  PUSH   HL
7E8B F5       Ø282Ø        PUSH   AF
7E8C C5       Ø283Ø        PUSH   BC
7E8D D5       Ø284Ø        PUSH   DE
7E8E 57       Ø285Ø        LD     D,A
7E8F CDAE7E   Ø286Ø        CALL   OUT
7E92 ØØ       Ø287Ø        NOP
7E93 ØØ       Ø288Ø        NOP                   ;TIMING
7E94 ØEØ8     Ø289Ø        LD     C,8
7E96 7A       Ø29ØØ E1     LD     A,D
7E97 Ø7       Ø291Ø        RLCA
7E98 57       Ø292Ø        LD     D,A
7E99 3ØØB     Ø293Ø        JR     NC,F
7E9B CDAE7E   Ø294Ø        CALL   OUT
7E9E ØD       Ø295Ø G      DEC    C
7E9F 2ØF5     Ø296Ø        JR     NZ,E1
7EA1 D1       Ø297Ø        POP    DE
7EA2 C1       Ø298Ø        POP    BC
7EA3 F1       Ø299Ø        POP    AF
7EA4 E1       Ø3ØØØ        POP    HL
7EA5 C9       Ø3Ø1Ø        RET
7EA6 Ø649     Ø3Ø2Ø F      LD     B,49H
7EA8 1ØFE     Ø3Ø3Ø H1     DJNZ   H1
7EAA ØØ       Ø3Ø4Ø        NOP
7EAB ØØ       Ø3Ø5Ø        NOP
7EAC 18FØ     Ø3Ø6Ø        JR     G
7EAE 3EØ5     Ø3Ø7Ø OUT    LD     A,5
7EBØ D3FF     Ø3Ø8Ø        OUT    (ØFFH),A
7EB2 Ø6Ø4     Ø3Ø9Ø        LD     B,Ø4H
7EB4 1ØFE     Ø31ØØ B1     DJNZ   B1
7EB6 3EØ6     Ø311Ø        LD     A,6
7EB8 D3FF     Ø312Ø        OUT    (ØFFH),A
7EBA Ø6Ø4     Ø313Ø        LD     B,Ø4H
7EBC 1ØFE     Ø314Ø C1     DJNZ   C1
7EBE 3EØ4     Ø315Ø        LD     A,4
7ECØ D3FF     Ø316Ø        OUT    (ØFFH),A
7EC2 Ø638     Ø317Ø        LD     B,38H
7EC4 1ØFE     Ø318Ø A1     DJNZ   A1
7EC6 C9       Ø319Ø        RET
7EC7 C5       Ø343Ø BLANK  PUSH   BC
7EC8 E5       Ø344Ø        PUSH   HL
7EC9 3E2Ø     Ø345Ø        LD     A,2ØH          ;BLANK
7ECB 21D67F   Ø346Ø        LD     HL,NAME
7ECE Ø6Ø6     Ø347Ø        LD     B,6
7EDØ 77       Ø348Ø BW1    LD     (HL),A
7ED1 23       Ø349Ø        INC    HL
7ED2 1ØFC     Ø35ØØ        DJNZ   BW1
7ED4 E1       Ø351Ø        POP    HL
7ED5 C1       Ø352Ø        POP    BC
7ED6 C9       Ø353Ø        RET
```

Listing continues

Listing continued

```
7ED7 AF         03540 GET    XOR    A
7ED8 32E57F     03550        LD     (IMM),A          ;RUN IMMEDIATELY
7EDB CDA77C     03570 BR     CALL   CK
7EDE 3E06       03580 BRI    LD     A,6              ;CODE FOR 'LD  B,N'
7EE0 32987F     03590        LD     (SWITCH),A
7EE3 21003C     03592        LD     HL,3C00H
7EE6 222040     03594        LD     (4020H),HL
7EE9 21DC7F     03600        LD     HL,TYPE
7EEC B8         03610        CP     B                ;B<6 IF NAME IS SPECIFIED
7EED 2005       03620        JR     NZ,BR6
7EEF 3EC9       03630        LD     A,0C9H           ;CODE FOR  'RET'
7EF1 32987F     03640        LD     (SWITCH),A
7EF4 CD217D     03650 BR6    CALL   SYNC             ;FIND SYNC CODE
7EF7 3E2A       03660        LD     A,2AH
7EF9 323E3C     03670        LD     (3C3EH),A
7EFC 323F3C     03680        LD     (3C3FH),A        ;ASTERISKS
7EFF 3E20       03690        LD     A,20H
7F01 ED5B2040   03710        LD     DE,(4020H)       ;CURSOR POSITION
7F05 12         03711        LD     (DE),A
7F06 13         03713        INC    DE
7F07 D5         03715        PUSH   DE
7F08 0606       03720        LD     B,6
7F0A CDA67F     03730 BR1    CALL   READ
7F0D 12         03744        LD     (DE),A
7F0E 13         03746        INC    DE
7F0F 10F9       03750        DJNZ   BR1
7F11 ED532040   03752        LD     (4020H),DE
7F15 D1         03755        POP    DE
7F16 CD987F     03760        CALL   SWITCH
7F19 0603       03770        LD     B,3
7F1B CDA67F     03780 BR2    CALL   READ
7F1E 77         03790        LD     (HL),A
7F1F 23         03800        INC    HL
7F20 10F9       03810        DJNZ   BR2
7F22 CDA67F     03820 BR3    CALL   READ
7F25 FE23       03830        CP     23H              ;END CODE
7F27 2830       03840        JR     Z,BRC
7F29 FE3C       03850        CP     3CH              ;BLOCK HEADER CODE
7F2B 20F5       03860        JR     NZ,BR3
7F2D CDA67F     03870        CALL   READ
7F30 FE3D       03872        CP     3DH              ;BLOCK HEADER CODE
7F32 20C0       03874        JR     NZ,BR6
7F34 CDA67F     03876        CALL   READ
7F37 47         03880        LD     B,A
7F38 CDA67F     03890        CALL   READ
7F3B 6F         03900        LD     L,A
7F3C CDA67F     03910        CALL   READ
7F3F 67         03920        LD     H,A
7F40 0E00       03930        LD     C,0
7F42 CD737F     03940 BRP    CALL   BRR
7F45 CDA67F     03950 BR4    CALL   READ
7F48 77         03960 BR7    LD     (HL),A
7F49 23         03970        INC    HL
7F4A 81         03980        ADD    A,C
7F4B 4F         03990        LD     C,A
7F4C 10F7       04000        DJNZ   BR4
7F4E CDA67F     04010        CALL   READ
7F51 B9         04020        CP     C
7F52 203C       04030        JR     NZ,ERR
7F54 CD2C02     04040        CALL   22CH             ;BLINK ASTERISKS
7F57 18C9       04050        JR     BR3
7F59 2B         04060 BRC    DEC    HL
7F5A CDF801     04070        CALL   1F8H             ;TURN OFF CASSETTE
7F5D 3ADC7F     04080        LD     A,(TYPE)
7F60 FE01       04090        CP     1
7F62 2010       04100        JR     NZ,BRG
7F64 22F940     04110        LD     (40F9H),HL
7F67 21E942     04120        LD     HL,42E9H
7F6A 22A440     04130        LD     (40A4H),HL
7F6D 3E0D       04140        LD     A,0DH
7F6F CD3300     04150        CALL   33H
7F72 D9         04160 BR5    EXX
7F73 C9         04170 BRR    RET
7F74 3AE57F     04180 BRG    LD     A,(IMM)
7F77 FE00       04190        CP     0
7F79 2803       04200        JR     Z,BRH
7F7B CDBA7D     04210 BRJ    CALL   EN               ;FIND ENTRY POINT
7F7E 2ADD7F     04220 BRH    LD     HL,(ENTRY)
7F81 3E01       04230        LD     A,1
7F83 32E57F     04240        LD     (IMM),A
7F86 228A7F     04250        LD     (BRF+1),HL
7F89 C30000     04260 BRF    JP     0
7F8C E1         04270 BRB    POP    HL
7F8D C3F47E     04280        JP     BR6              ;WRONG NAME FIND NEW SYNC
7F90 214345     04290 ERR    LD     HL,4543H
7F93 223C3C     04300        LD     (3C3CH),HL
7F96 188A       04310        JR     BR3
7F98 0606       04320 SWITCH LD     B,6              ;RETURN IF NO NAME
7F9A 21D67F     04330        LD     HL,NAME
7F9D 1A         04340 BRA    LD     A,(DE)
7F9E BE         04350        CP     (HL)
7F9F 20EB       04360        JR     NZ,BRB
7FA1 23         04370        INC    HL
7FA2 13         04380        INC    DE
7FA3 10F8       04390        DJNZ   BRA
7FA5 C9         04400        RET                     ;RIGHT NAME FOUND
7FA6 3A4038     04410 READ   LD     A,(3840H)
7FA9 FE04       04412        CP     4
7FAB CA191A     04414        JP     Z,1A19H
7FAE E5         04416        PUSH   HL
7FAF C5         04420        PUSH   BC
7FB0 3E04       04430        LD     A,4
7FB2 D3FF       04440        OUT    (0FFH),A
7FB4 DBFF       04450 P1     IN     A,(0FFH)
7FB6 17         04460        RLA
7FB7 30FB       04470        JR     NC,P1
7FB9 0621       04480        LD     B,21H
7FBB 10FE       04490 R1     DJNZ   R1
7FBD 0608       04500        LD     B,8
7FBF 3E04       04510 S      LD     A,4
7FC1 D3FF       04520        OUT    (0FFH),A
7FC3 60         04530        LD     H,B
7FC4 0626       04540        LD     B,26H
7FC6 DBFF       04550 Q      IN     A,(0FFH)
7FC8 10FC       04560        DJNZ   Q
7FCA 17         04570        RLA
7FCB CB15       04580 W      RL     L
7FCD 44         04590        LD     B,H
7FCE 10EF       04600        DJNZ   S
7FD0 7D         04610        LD     A,L
7FD1 C1         04620        POP    BC
7FD2 E1         04630        POP    HL
7FD3 C9         04640        RET
7FD4 A5A6       04650 HEAD   DEFW   0A6A5H
0006            04660 NAME   DEFS   6
0001            04670 TYPE   DEFS   1
0002            04680 ENTRY  DEFS   2
0002            04690 COUNT  DEFS   2
0002            04700 START  DEFS   2
0002            04710 END    DEFS   2
7FE5 01         04720 IMM    DEFB   1
7FE6 53         04740 ST     DEFM   'START '
7FEC 00         04750        DEFB   0
7FED 20         04760 ED     DEFM   ' END '
7FF2 00         04770        DEFB   0
7FF3 20         04780 ENT    DEFM   ' ENTRY '
7FFA 00         04790        DEFB   0
7FFB 42         04800 BAD    DEFM   'BAD '
7FFF 00         04810        DEFB   0
3C54            04830        ORG    3C54H
3C54 47         04832        DEFM   'GOOD LOAD
1A19            04834        END    1A19H
00000 TOTAL ERRORS
```

COMPUTER BOOKS FOR BEGINNERS

Everything you need to get started programming your own personal computer. These handy books of programs, and about programming, are loaded with easy-to-understand info for beginners. The books include hundreds of ready-to-type-and-run programs as well as hundreds of program-writing tips, tricks, hints, shortcuts, secrets, techniques. We offer books covering the 13 most popular computers for beginners: TRS-80 Color Computer, Atari 400, Atari 800, Apple II, Sinclair ZX-81, Timex 1000, Micro Ace, IBM Personal Computer, Casio FX-702P pocket computer, Sharp PC-1211 and PC-1500 pocket computers and TRS-80 PC-1 and PC-2 pocket computers.

TRS-80/Sharp/Casio Pocket Computers

99 Tips & Tricks for the New Pocket Computers, all new programs, using tremendous power of TRS-80, PC-2/Sharp PC-1500, LCD graphics, printer/plotter graphics, useful business and home software, includes 99 complete type-and-run programs, learn full range of expanded BASIC, 128 pages **$7.95**

Pocket Computer Programming Made Easy, fast new, easy read-and-learn way to quickly understand the BASIC programming language, how to make TRS-80 PC-1/PC-2, Sharp PC-1211/PC-1500 & Casio FX-702P computers work for you, 128 pages **$8.95**

101 Pocket Computer Programming Tips & Tricks, secrets, hints, shortcuts, techniques from a master programmer, includes 101 ready-to-run programs, for TRS-80 PC-1 and PC-2 and Sharp PC-1211 and PC-1500, 128 pages **$7.95**

50 Programs in BASIC for Home, School & Office, useful ready-to-run software for PC-1/PC-2/PC-1211/PC-1500, 96 pages **$9.95**

50 MORE Programs in BASIC for Home, School & Office, book of tested software for PC-1/PC-2/PC-1211/PC-1500, 96 pages **$9.95**

Murder In The Mansion and Other Computer Adventures, mystery, space adventure, games, 24 programs for PC-1/PC-2/PC-1211/PC-1500, 96 pages **$6.95**

35 Practical Programs for the Casio Pocket Computer, useful type- and run software for the FX-702P, 96 pages **$8.95**

Atari 400/800 Computers

101 ATARI Computer Programming Tips & Tricks, learn-by-doing instruction, hints, secrets, shortcuts, techniques for Atari 400 and 800 computers, includes 101 ready-to-run programs, 128 pages **$8.95**

31 New ATARI Computer Programs for Home, School & Office, practical type-and-run software for Atari 400 and 800, 96 pages **$8.95**

Timex 1000/Sinclair ZX-81

101 TIMEX 1000/Sinclair ZX-81 Programming Tips & Tricks, secrets, hints, shortcuts, learn-by-doing instruction, techniques for the ZX-81, MicroAce and Timex 1000 computers, includes 101 ready-to-run programs, 128 pages **$7.95**

37 TIMEX 1000/Sinclair ZX-81 Computer Programs for Home, School & Office, practical type-and-run software for ZX-81, Timex 1000 and MicroAce, 96 pages **$8.95**

TRS-80 Color Computer

Color Computer Graphics, complete guide loaded with instruction, how to make the most of TRS-80 Color Computer video graphics, many complete programs, 128 pages **$9.95**

The Color Computer Songbook, 40 favorite pop, folk, classical, seasonal songs arranged for play on TRS-80 Color Computer, type-and-run music programs, 96 pages **$7.95**

101 Color Computer Programming Tips& Tricks, learn-by-doing instructions, hints, secrets, techniques, insights, for TRS-80 Color Computer, includes 101 programs, 128 pages **$7.95**

55 Color Computer Programs for Home, School & Office, practical ready-to-run software, colorful graphics, for TRS-80 Color Computer, 128 pages **$9.95**

55 MORE Color Computer Programs for Home, School & Office, handy companion volume packed with different useful type-and-run software, colorful graphics, for TRS-80 Color Computer, 112 pages **$9.95**

My Buttons Are Blue and Other Love Poems from the Digital Heart of An Electronic Computer, for poetry lovers, computer fans, a high-tech classic, 66 heartwarming poems written by a TRS-80 Color Computer, 96 pages **$4.95**

Apple Computer

101 APPLE Computer Programming Tips & Tricks, secrets, hints, insights, 101 ready-to-run programs for Apple II, 128 pages **$8.95**

33 New APPLE Computer Programs for Home, School & Office, practical type-and-run software for Apple II, 96 pages **$8.95**

Program Worksheets

Tablets of handy printed forms make writing BASIC software easy and fun. Customized for computer systems, or use the universal form good for any BASIC computer. 40-sheet pads.

Color Computer BASIC Coding Form	**$2.95**
Pocket Computer BASIC Coding Form	**$2.95**
APPLE Computer BASIC Coding Form	**$2.95**
TIMEX/Sinclair BASIC Coding Form	**$2.95**
IBM Personal Computer Coding Form	**$2.95**
ATARI Computer BASIC Coding Form	**$2.95**
Universal BASIC Coding Form	**$2.95**

Order direct from this ad. Send check, money order, or MasterCard or VISA account number and expiration date. Include $1 shipping for each item ordered up to a maximum of $3. Or write for our free catalog. Mail to:

ARCsoft Publishers

Post Office Box 132 L
Woodsboro, Maryland 21798

(301) 663-4444

Customers wanting airmail send $4 postage per item ordered. Foreign customers pay in U.S. dollars. Maryland residents add 5% sales tax. Sorry, no COD. ✓6

✓See List of Advertisers on Page 563

The Glamour of Grammar

by George Stone

So your students don't appreciate the finer points of grammar? This program will let them learn the rules and have fun at the same time.

Kids are fascinated by video displays. They enjoy telling the computer to do something, then watching it get done. I wrote Sentence Patterns with the hope that this fascination would get my students interested in learning grammar.

The program drills students on sentence patterns by mimicking what I do in the classroom. First, it makes up a sentence. Then, it asks directed questions about the sentence to lead the kids to the sentence pattern. By using this program, the students learn the sequence of thoughts needed to determine sentence patterns.

Why even learn sentence patterns? To pass the test, of course. Besides that, knowing how to find the pattern helps in understanding the meaning of long, involved sentences found in literature, especially poetry. It also helps my students proofread their writing for complete sentences.

Computers Are Motivating

The results met my expectations. But then other factors popped up that proved even more motivating than the video appeal. The kids see the questions as a challenge, a dare. They don't like being outsmarted by the machine, so they think twice and try harder for the correct answer.

They also enjoy the absurd nature of the sentences being presented (more about this later). To see the next weird sentence, they have to answer all the questions about the one being displayed. Not only that, but the higher the level (i.e., the more complicated the sentence), the more absurd the sentence becomes. Since the students can't advance to higher levels without passing (above 85 percent) the lower ones, this encourages achievement.

But the biggest motivator of all turned out to be the kids themselves. When I let two or three kids at a time use the computer, they taught each other. They argued and haggled and debated over which answer to choose. They were giving reasons. They were using rules I had taught in class to defend their answers. They were really learning. All because of a few words displayed on a screen.

Although I didn't have strict control or experimental groups, the kids who used the computer more often averaged higher scores on the test than the ones who didn't.

Evolution of the Program

When I was learning Basic, two features appealed to me: the random-number generator and string variables. Since grammar is logic imposed on lan-

SENTENCE GENERATOR
LEVELS OF COMPLEXITY

Line number	Level	
209	8	Complex sentence using a level 4 dependent clause and a level 3 main clause
210	7	Complex sentence using a level 3 dependent clause and a level 4 main clause
211	6	Complex sentence using a level 3 dependent clause and a level 3 main clause
212	5	Complex sentence using a level 2 dependent clause and a level 3 main clause
215	4	Level 3 plus an adjective phrase or clause
220	3	Level 2 plus another adjective and a prepositional phrase
225	2	Level 1 plus an adjective and an adverb
230	1	Simple sentence pattern with no modifiers

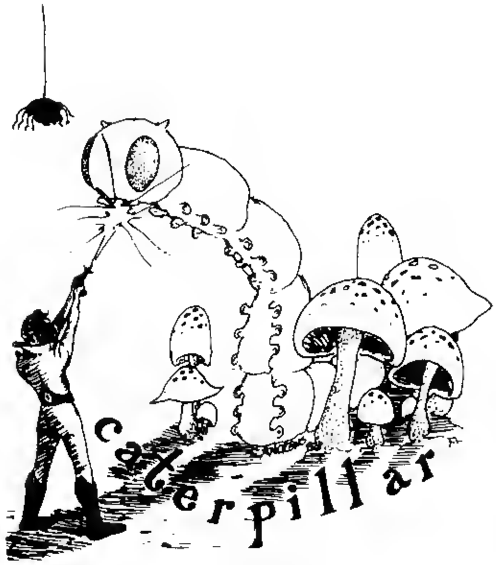
Table 1. Levels of complexity

The Key Box

Model I or III
16K RAM
Cassette or Disk Basic
(POKE 16396, 165 on the Model I to disable break.)

AARDVARK

TRS-80 COLOR OSI VIC-64 VIC-20 SINCLAIR TIMEX



TUBE FRENZY
(by Dave Edson)
This is an almost indescribably fast action arcade game. It has fast action, an all new concept in play, simple rules, and 63 levels of difficulty. All machine code, requires Joysticks. Another great game by Dave Edson. TRS 80 COLOR ONLY. 16k and Joysticks required. $19.95.

QUEST — A NEW IDEA IN ADVENTURE GAMES! Different from all the others. Quest is played on a computer generated map of Alesia. Your job is to gather men and supplies by combat, bargaining, exploration of ruins and temples and outright banditry. When your force is strong enough, you attack the Citadel of Moorlock in a life or death battle to the finish. Playable in 2 to 5 hours, this one is different every time. 16k TRS-80, TRS-80 Color, and Sinclair. 13K VIC-20. $14.95 each.

ADVENTURES!!!
These Adventures are written in BASIC, are full featured, fast action, full plotted adventures that take 30-50 hours to play. (Adventures are interactive fantasies. It's like reading a book except that you are the main character as you give the computer commands like "Look in the Coffin" and "Light the torch.")

Adventures require 16k on TRS80, TRS80 color, and Sinclair. They require 8k on OSI and 13k on Vic-20. Derelict takes 12k on OSI. $14.95 each.

CATERPILLAR
O.K., the Caterpillar does look a lot like a Centipede. We have spiders, falling fleas, monsters traipsing across the screen, poison mushrooms, and a lot of other familiar stuff. COLOR 80 requires 16k and Joysticks. This is Edson's best game to date. $19.95 for TRS 80 COLOR.

PROGRAMMERS!
SEE YOUR PROGRAM IN THIS SPACE!!
Aardvark traditionally pays the highest commissions in the industry and gives programs the widest possible coverage. Quality is the keyword. If your program is good and you want it presented by the best, send it to Aardvark.

ESCAPE FROM MARS
(by Rodger Olsen)
This ADVENTURE takes place on the RED PLANET. You'll have to explore a Martian city and deal with possibly hostile aliens to survive this one. A good first adventure.

PYRAMID (by Rodger Olsen)
This is our most challenging ADVENTURE. It is a treasure hunt in a pyramid full of problems. Exciting and tough!

HAUNTED HOUSE (by Bob Anderson)
It's a real adventure—with ghosts and ghouls and goblins and treasures and problems — but it is for kids. Designed for the 8 to 12 year old population and those who haven't tried Adventure before and want to start out real easy.

DERELICT
(by Rodger Olsen & Bob Anderson)
New winner in the toughest adventure from Aardvark sweepstakes. This one takes place on an alien ship that has been deserted for a thousand years — and is still dangerous!

CATCH'EM
(by Dave Edson)
One of our simplest, fastest, funnest, all machine code arcade games. Raindrops and an incredibe variety of other things come falling down on your head. Use the Joysticks to Catch'em. It's a BALL! — and a flying saucer! — and a Flying Y!— and so on. TRS 80 COLOR. $19.95.

BASIC THAT ZOOOMMS!!
AT LAST AN AFFORDABLE COMPILER!
The compiler allows you to write your programs in easy BASIC and then automatically generates a machine code equivalent that runs 50 to 150 times faster.

It does have some limitations. It takes at least 8k of RAM to run the compiler and it does only support a subset of BASIC—about 20 commands including FOR, NEXT, END, GOSUB, GOTO, IF, THEN, RETURN, END, PRINT, STOP, USR (X), PEEK, POKE, *, /, +, -, >, <, =, VARIABLE NAMES A-Z, SUBSCRIPTED VARIABLES, and INTEGER NUMBERS FORM 0-64K.

TINY COMPILER is written in BASIC. It generates native, relocatable 6502 or 6809 code. It comes with a 20-page manual and can be modified or augmented by the user. $24.95 on tape or disk for OSI, TRS-80 Color, or VIC.

Please specify system on all orders

ALSO FROM AARDVARK — **This is only a partial list of what wa carry. Wa hava a lot of other games (particularly for tha TRS-80 Color and OSI), business programs, blank tapes and disks and hardwara. Send $1.00 for our complete catalog.**

VISA

AARDVARK - 80
2352 S. Commerce, Walled Lake, MI 48088
(313) 669-3110

master charge

Phone Orders Accepted 8:00 a.m. to 4:00 p.m. EST. Mon.-Fri.

107

guage, why couldn't a computer start with the logic and insert the language?

Time out for some grammar review. Any sentence can be classified as one of four patterns: subject-verb, subject-verb-direct object, subject-linking verb-adjective, or subject-linking verb-noun. (Some grammar systems are more complex, but for my students this is enough.)

For example, in a sentence like, "Suzy is pretty," the sentence pattern is S-LV-Adj. "Is" links "Suzy" to "pretty." Almost any noun can be substituted for "Suzy," almost any linking verb can be substituted for "is," and almost any adjective can take the place of "pretty." Example: "The hunchback looked ugly." (Someday the Basic software people will invent an If...Then...Except operation for language strings.)

The point is that the computer can create its own sentences by picking subjects, verbs, adjectives, and other parts of speech from arrayed word lists, then arranging them according to English syntax rules.

Making up sentences this way has its faults and advantages. Since only a part of a sentence's meaning comes from its structure, and since the computer (obviously) has no idea what

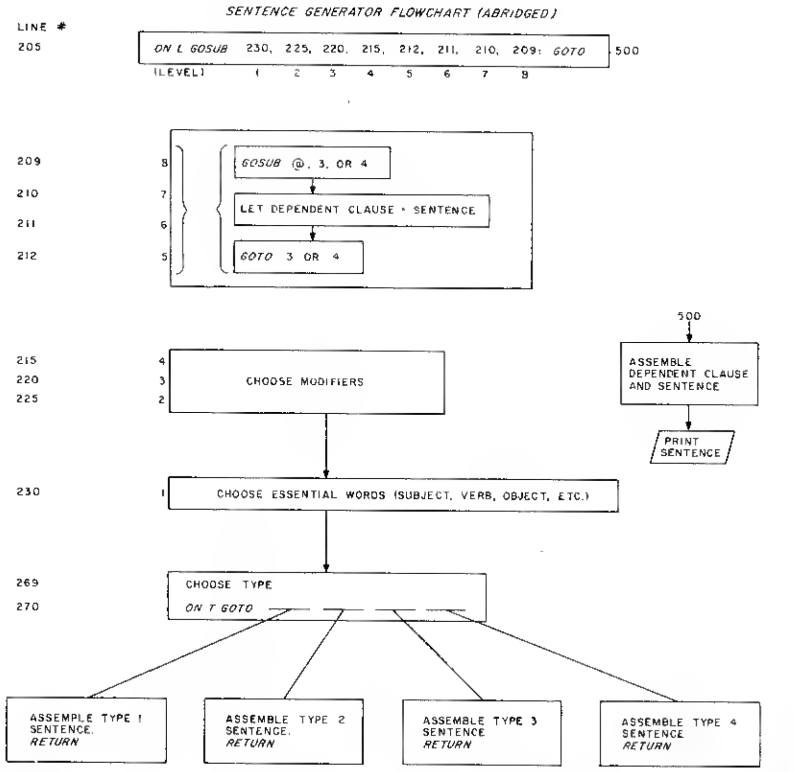


Fig. 1. The return works two ways, depending on where the GOSUB was initiated. On levels 1–4, the return returns to line 205. On levels 5–8, the return is nested and returns first to the level 5–8 lines, then to 205.

words it's using, it can create such monstrosities as:

"The cute dog smelled a robot on the lawn since a weird girl later tricked the fish around the tree."

"Because Albert, too tired to laugh, foolishly smelled a fat man around the tree, a fat mouse, feeling gay, finally threw a foolish mouse on the lawn."

As it turned out, the kids were entertained by the bizarre and fantastical nature of these sentences. The absurdity of the meaning seems to make the pattern a little more obvious.

As I was developing the sentence generator, it occurred to me that since the program started with the answer (the sentence pattern) and came up with an example, why not turn this into a learning exercise? Hence the question-and-answer part of the program.

Program Explanation

The sentence-generating part of the program, lines 1–91 and lines 200–500, creates four types (patterns) of sentences at eight different levels of difficulty. (See Table 1.)

The levels of difficulty are possible because when a string is never defined, it will not cause an error when concatenated in a longer string. The null string is ignored. This allows the program to "choose" which words it will or will not use by simply putting a number or a zero in the subscript.

For example, AJ$(0) is null but AJ$(1) is "fat." Level 1 sentences use only the key words in the sentence pattern. The modifiers are left out by giving them all zero subscripts. But level 3 sentences have adjectives, adverbs, and prepositional phrases.

To build a complex sentence (levels 5–8), the program assembles a sentence (CS$), adds a subordinating conjunction (see data line 21; I'm not going to explain subordinating conjunctions), calls it a dependent clause [DC$ = D$(D) + CS$], then passes through the building sequence again to create a main clause. This all starts in line 200, and the two clauses are concatenated in line 501.

Lines 504 to 515 keep the words from wrapping around the screen by breaking the sentence up into pieces that will fit on a line. It's messy, but it works.

The sentence is first printed in the loop in line 518. It is printed on every other line to leave space for subroutine 6000 (more on this later).

The question-and-answer part of the program runs between 520 and 999. For each question, the question (Q$),

	String Variables
A$	Student's answer
AA$( )	Correct answer statement
AJ$(AJ)	Adjective list
B1$, B2$	Graphics for eater's bottom half
CA$	Correct answer
CS$	Complete sentence or main clause
CS$( )	Chunks of CS$ less than 64 characters long
D$(D)	Subordinating conjunction list
DC$	Dependent clause
F1$( ), F2$( )	Graphics for word eater's mouth
LV$(LV)	Linking verb list
ND$(ND)	Noun determiner list
NME$	Student's name
NO$(NO)	Object noun list
NS$	Subject of current sentence
NS$(NS)	Subject noun list
O$	Used after INPUT
P$	Predicate complement of current sentence
PA$(PA)	Predicate adjective list
PH$(PH)	Adjective phrase or clause
PN$(PN)	Predicate noun list
PP$(PP)	Prepositional phrase list
Q$	Current question
QC$	Question clarifier
QYN$	Answer choices for yes/no type of questions
S$( )	Stores names of students
U$, Z$, ZL$, ZR$	Graphics for worm
V$	Verb of current sentence
VI$(VI)	Intransitive verb list
VT$(VT)	Transitive verb list
	Numerical Variables
L	Level
LN	LEN(CS$)
LS	Number of chunks CS$ is broken into
P	Percent of right answers
Q	Question number
R	Number of questions right
S	Number of students who have used the computer
SL( )	Stores student's level
SP( )	Stores percent of right answers
ST( )	Stores total (T1)
T1	Number of questions attempted
TP( )	Stores total number of points (L × R)
U	PEEK variable
W	Timing loop
M, N, X, Y	Loop variables

Table 2

Program Listing

```
1 CLS:PRINT@215,"SENTENCE PATTERNS:";CHR$(232);"AN INTERACTIVE L
EARNING PROGRAM";:PRINT@478,"BY";CHR$(249);"GEORGE STONE";CHR$(2
45);"MAY, 1982"
8 CLEAR1200:RANDOM:T1=0:R=0:DIMNS$(15):POKE16396,165
9 DATA" THE"," A"
10 DATA " HORSE"," DOG"," HIPPO"," MONKEY"," FERRET"," MAN"," BO
Y"," GIRL"," MOUSE"," TEACHER"," MYRA"," HERMAN"," ALBERT"," GER
ALDINE"," SUZYCREAMCHEESE"
11 DATA" BELCHED"," SAT"," LEANED"," RAN"," CRAWLED"," WALKED","
 LOOKED"," SQUIRMED"," YAWNED"," DANCED"
12 DATA " THREW"," ATE"," SAW"," SMELLED"," TRICKED"
13 DATA " ROBOT"," MAN"," MOUSE"," RHINOCEROS"," FISH"
14 DATA " FAT"," FOOLISH"," CURIOUS"," WEIRD"," CUTE"," CRAZY","
 SMART"," LAZY"," SLEEPY"," BEAUTIFUL"
15 DATA " NEAR THE TREE"," OVER THE HILL"," ACROSS THE FIELD","
UNDER WATER"," ON THE LAWN"
16 DATA ", TOO TIRED TO LAUGH,",", FEELING GAY,"," WHO DAVE LIKE
D"," WHO JUST RETURNED FROM THE MOON",", HOPELESSLY CONFUSED,"
```

Listing continues

Listing continued

```
17 DATA " MERRILY"," FOOLISHLY"," IMMEDIATELY"," FINALLY"," LATE
R"," SUDDENLY"," QUICKLY"," USUALLY"
18 DATA" WAS"," BECAME"," IS"," WAS"," HADBEEN"," LOOKED"," SEEM
ED"," FELT"," BECAME"," WAS"
19 DATA" JOCK"," LOUDMOUTH"," RATFINK"," FOOL"," CLOWN"
20 DATA" GOOD"," BAD"," LONELY"," FINE"," NERVOUS"," HAPPY"," BL
UE"," STINKY"," GAWKY"," BEAUTIFUL"
21 DATA" IF"," WHEN"," SINCE"," AS"," BECAUSE"," SINCE"," WHILE"
," ALTHOUGH"," BEFORE"," BECAUSE"
50 F1$(1)="":F2$(1)=CHR$(190)+STRING$(2,188)+STRING$(2,176)+STRI
NG$(2,128)+STRING$(2,176)+STRING$(2,188)+CHR$(189)
55 F1$(2)=CHR$(176)+CHR$(180)+STRING$(6,128)+CHR$(184)+CHR$(176)
:F2$(2)=CHR$(190)+STRING$(3,191)+CHR$(188)+CHR$(144)+CHR$(160)+C
HR$(188)+STRING$(3,191)+CHR$(189)
60 F1$(3)=CHR$(176)+CHR$(180)+STRING$(6,128)+CHR$(184)+CHR$(176)
:F2$(3)=CHR$(190)+STRING$(3,191)+CHR$(188)+CHR$(144)+CHR$(160)+C
HR$(188)+STRING$(3,191)+CHR$(189)
65 F1$(4)=CHR$(176)+STRING$(2,188)+STRING$(4,191)+STRING$(2,188)
+CHR$(176):F2$(4)=CHR$(190)+STRING$(10,191)+CHR$(189)
70 B1$=CHR$(175)+STRING$(9,191)+CHR$(188)+CHR$(159):B2$=CHR$(128
)+CHR$(131)+STRING$(2,143)+STRING$(4,191)+STRING$(2,143)+CHR$(13
1)+CHR$(128)
75 Z$=STRING$(7,176):ZL$=CHR$(140)+CHR$(176)+CHR$(158)+CHR$(131)
+CHR$(173)+CHR$(176)+CHR$(194)
76 U$=CHR$(195)+CHR$(191)+CHR$(195):ZR$=STRING$(4,24)+CHR$(199)+
CHR$(176)+CHR$(158)+CHR$(131)+CHR$(173)+CHR$(176)+CHR$(140)
79 FORN=1TO2:READND$(N):NEXT
80 FORN=1TO15:READNS$(N):NEXT
81 FORN=1TO10:READVI$(N):NEXT
82 FORN=1TO5:READVT$(N):NEXT
83 FORN=1TO5:READNO$(N):NEXT
84 FORN=1TO10:READAJ$(N):NEXT
85 FORN=1TO5:READPP$(N):NEXT
86 FORN=1TO5:READPH$(N):NEXT
87 FORN=1TO8:READAV$(N):NEXT
88 FORN=1TO10:READLV$(N):NEXT
89 FORN=1TO5:READPN$(N):NEXT
90 FORN=1TO10:READPA$(N):NEXT
91 FORN=1TO10:READD$(N):NEXT
149 PRINT@960,:FORN=1TO9:PRINT:FORW=1TO70:NEXT:NEXT
150 CLS:PRINT@256,"(<ENTER> FOR SCORES)";:PRINT@5,"S E N T E N C
 E   P A T T E R N   P R A C T I C E":PRINT:NME$="":INPUT"WHAT I
S YOUR NAME";NME$
155 IFNME$=""THEN160ELSEFORN=1TOS:IFLEFT$(NME$,15)=S$(N)THEN170E
LSENEXT:GOTO181
160 CLS:PRINT"  NAME","LEVEL","SCORE","TOTAL POINTS":FORN=1TOS:P
RINTS$(N),SL(N),SP(N),TP(N):NEXT
165 PRINT@910,"HIT <ENTER>":INPUTO$:GOTO150
170 T1=ST(N):L=SL(N):R=INT(T1*SP(N)/100):PRINT"I REMEMBER YOU, "
;NME$:PRINT:PRINT"DO YOU WANT YOUR OLD SCORE   (Y/N)";:GOSUB1000
:INPUTO$
175 IFO$="N"THENR=0:T1=0:GOTO186ELSECLS:GOTO200
181 PRINT:PRINT"DO YOU WANT DIRECTIONS,  ";NME$;",  (YES/NO)";:I
NPUTO$:IFLEFT$(O$,1)="N"THEN186
182 PRINT"DIRECTIONS:":PRINT:PRINT"     A SENTENCE WILL BE DISPL
AYED AT THE TOP OF THE SCREEN.     YOUR JOB, SHOULD YOU CHOOSE T
O ACCEPT IT, IS TO CORRECTLY ANSWERTHE QUESTIONS ABOUT THAT SENT
ENCE."
183 PRINT:PRINT"     TO ANSWER A QUESTION, MOVE THE ARROW (";CHR
$(91);") BACK AND FORTH     USING THE RIGHT AND LEFT ARROW KEYS
ON THE KEYBOARD":PRINTTAB(18)"THEN HIT <ENTER>"
186 L=0:INPUT"WHAT LEVEL DO YOU WANT?(1-6)";L
187 IF(L>6)+(L<1)THENCLS:PRINTCHR$(23):PRINT"NO, NO, NO, YOU IDI
OTIC IDIOT!!    ":PRINT"**** A NUMBER BETWEEN 1 (ONE)     AND 6
(SIX)":PRINT:PRINT"#%&'($&()%$#!!!$&%'&(&%)'$$##":PRINT:GOTO186
200 DC$="":D=RND(10):AJ2=0:AJ=0:PP=0:PH=0:AV=0:CLS:PRINT"HOLD ON
 WHILE I THINK UP A GOOD ONE"
205 ONLGOSUB230,225,220,215,212,211,210,209:GOTO500
209 GOSUB215:DC$=D$(D)+CS$:GOTO215
210 GOSUB220:DC$=D$(D)+CS$:GOTO215
211 GOSUB220:DC$=D$(D)+CS$:GOTO220
212 GOSUB225:DC$=D$(D)+CS$:GOTO220
215 PH=RND(5)
220 PP=RND(5):AJ2=RND(10)
225 AJ=RND(10):AV=RND(8)
230 ND=RND(2):DN=RND(2):NS=RND(15):VI=RND(10):VT=RND(5):NO=RND(5
):LV=RND(5):PN=RND(5):PA=RND(10)
235 IFNS>10THENND=0:AJ=0
269 T=RND(4):IFT=3THENLV=LV+5
270 ONTGOTO300,350,400,450
300 REM   ***    S---V-
310 CS$=ND$(ND)+AJ$(AJ)+NS$(NS)+PH$(PH)+AV$(AV)+VI$(VI)+PP$(PP)
315 NS$=MID$(NS$(NS),2):V$=MID$(VI$(VI),2):P$="":RETURN
350 REM   ***    S--V--DO
360 CS$=ND$(ND)+AJ$(AJ)+NS$(NS)+PH$(PH)+AV$(AV)+VT$(VT)+ND$(DN)+
```

Listing continues

question clarifier (QC$, if there is one), correct answer (CA$), yes/no answer (QYN$, if there is one), and complete answer (AA$) are defined. Then the answer subroutine (3000) is called. Subroutine 4000 prints the sentence again, and the next set of questions is prepared for subroutine 3000.

Five or six questions are asked about each sentence, thus getting the students to look at each part of the sentence. The whole point of this exercise is in the last question (line 900). Using the information the program has suggested about the sentence, the students have to decide what the sentence pattern is.

The Subroutines

Subroutine 3000 prints all the previous correct answers, prints the current question, calls subroutine 6000 for the answer (A$), compares the answer with the correct answer, gives a prompt (QC$) if needed, and adds to the score if the answer is right. (Whew!)

Subroutine 1000 prints the cumulative score and current level.

Subroutine 1100 keeps track of who has been using the computer and sorts the students according to their scores. A screen print of this information can be accessed from the beginning of the program (lines 155–160) when it asks for the next student's name.

Subroutines 5000 and 5500 are graphics treats or tricks to relieve tedium and reinforce correct answers. They can be canceled by pressing the space bar.

Now to explain subroutine 6000. The first version of this program had the students typing in the answers. But they spent most of their time hunting and pecking rather than thinking about subjects and verbs. Having them give the first letter didn't work because sometimes the wrong word had the same first letter as the right one. Subroutine 6000 evolved after watching the advanced computer students design their own games.

Line 6020 prints an up arrow under the sentence. Lines 6030–6060 let the students move the arrow back and forth using the right and left-arrow keys. When they've found an answer, they press enter, which causes the program to jump to line 6100. Lines 6100–6180 PEEK the video memory locations above the arrow and then build an answer word (A$) based on the spaces between words. The formula in line 6180 resets the cursor to the space after the last question displayed.

Listing continued

```
PA$(AJ2)+NO$(NO)+PP$(PP)
365 NS$=MID$(NS$(NS),2):V$=MID$(VT$(VT),2):P$=MID$(NO$(NO),2):RE
TURN
400 REM   ***S-LV-A
410 CS$=ND$(ND)+AJ$(AJ)+NS$(NS)+PH$(PH)+LV$(LV)+PA$(PA)+PP$(PP)
415 NS$=MID$(NS$(NS),2):V$=MID$(LV$(LV),2):P$=MID$(PA$(PA),2):RE
TURN
450 REM   ***   S--LV--N
460 CS$=ND$(ND)+AJ$(AJ)+NS$(NS)+PH$(PH)+LV$(LV)+ND$(DN)+PA$(AJ2)
+PN$(PN)+PP$(PP)
470 NS$=MID$(NS$(NS),2):V$=MID$(LV$(LV),2):P$=MID$(PN$(PN),2):RE
TURN
500 Q=1:
501 IFLEN(DC$)>0ANDRND(2)=1THENCS$=DC$+","+CS$+"."ELSECS$=CS$+DC
$+"."
504 LN=LEN(CS$):IFLN>191THEN200
505 IFLN<64THENLS=1:CS$(1)=CS$:GOTO518
508 LS=2:FORM=63TO50STEP-1:IFMID$(CS$,M,1)=" "THENCS$(1)=LEFT$(C
S$,M):CS$(2)=MID$(CS$,M+1):GOTO510ELSENEXTM
510 N=LN+1:IFLN>M+63THENLS=3:FORN=M+63TOM+50STEP-1:IFMID$(CS$,N,
1)=" "THEN515ELSENEXTN
515 CS$(2)=MID$(CS$,M,N-M):CS$(3)=MID$(CS$,N)
518 CLS:FORN=1TOLS:PRINTCS$(N):PRINT:NEXT
520 Q$="  WHAT IS THE SUBJECT OF THIS SENTENCE?"
530 CA$=NS$:QC$="  WHO OR WHAT IS THIS SENTENCE ABOUT?":AA$(1)="
THE SUBJECT IS '"+NS$+"'"
540 GOSUB3000:GOSUB4000
600 CA$=V$:Q$="   WHAT IS THE VERB OF THIS SENTENCE?":QC$="  WHI
CH WORD SHOWS AN ACTION OR STATE OF BEING?":AA$(2)="THE VERB IS
'"+V$+"'"
610 GOSUB3000
620 GOSUB4000
650 Q$="   WHAT TYPE OF VERB IS '"+V$+"'?":QYN$="ACTION LINKING"
660 IFT<=2THENCA$="ACTION":AR$="N"
665 IFT>=3THENCA$="LINKING":AR$=""
670 AA$(3)="'"+V$+"' IS A"+AR$+" "+CA$+" VERB"
680 GOSUB3000:GOSUB4000
690 ONTGOTO700,700,800,800
700 Q$="  IS THERE A DIRECT OBJECT? ":QYN$="YES    NO"
705 IFT=1 THENCA$="NO":AA$(Q)="THERE IS NO DIRECT OBJECT":GOSUB3
000:GOSUB4000:GOTO750
710 IFT=2THENCA$="YES":AA$(Q)="THERE IS A DIRECT OBJECT"
720 GOSUB3000:GOSUB4000
730 Q$="WHAT IS THE DIRECT OBJECT?":QC$="WHAT WORD IS GETTING TT
HE ACTION OF THE VERB?":CA$=P$:AA$(Q)=P$+" IS THE DIRECT OBJECT"

740 GOSUB3000:GOSUB4000
750 GOTO900
800 Q$="WHAT IS '"+NS$+"' BEING LINKED TO?":QC$="IN OTHER WORDS,
 THE "+NS$+" "+V$+" WHAT?":CA$=P$:AA$(Q)="'"+P$+"' IS THE SUBJEC
T COMPLEMENT"
810 GOSUB3000:GOSUB4000
840 QYN$="NOUN   ADJ":Q$="IS '"+P$+"' A NOUN OR AN ADJECTIVE?"
850 IFT=4THENCA$="NOUN":AA$(Q)="'"+P$+"' IS A NOUN"
860 IFT=3THENCA$="ADJECTIVE":AA$(Q)="'"+P$+"' IS AN ADJECTIVE"
870 GOSUB3000:GOSUB4000
900 Q$="LAST QUESTION......WHAT IS THE SENTENCE PATTERN OF THIS
   SENTENCE?   (1)S-V    (2)S-V-DO  (3) S-LV-ADJ   (4) S-LV-NOUN
        PRESS A NUMBER"
910 QC$="":ONTGOTO912,914,916,918
912 CA$="1":AA$(Q)=" S-V":GOTO920
914 CA$="2":AA$(Q)=" S-V-DO":GOTO920
916 CA$="3":AA$(Q)=" S-LV-ADJ":GOTO920
918 CA$="4":AA$(Q)="S-LV-NOUN":GOTO920
920 AA$(Q)=AA$(Q)+"    "+NS$+"--"+V$+"--"+P$:PRINT@768,"";:GOSUB3
000
950 CLS:PRINT:PRINT:PRINT"     YOUR SCORE SO FAR IS ";INT(R/T1*1
00);"% ON LEVEL ";L:PRINT
952 IF(L>1)*(R/T1<.5)THENL=L-1:PRINT"YOU HAVE JUST MOVED DOWN TO
 LEVEL ";L;"  THIS SHOULD BE EASIER"
956 IF(L=8)*(R/T1>.95)THENCLS:FORN=1TO60:PRINT"!!WOW!!";(R*100/T
1);"%!!";:FORW=1TO30:NEXT:NEXT:CLS:PRINTCHR$(23):PRINT:PRINT:PRI
NTR*100/T1;"%  ON  LEVEL ";L:PRINT:PRINT"NICE WORK, ";NME$:PRINT
"  I'M DONE WITH YOU ":FORW=1TO1500:NEXT:O$="N":GOTO970
957 IF(L<8)*(R/T1>.85)THENL=L+1:PRINT"YOU HAVE JUST MOVED UP TO
LEVEL ";L
960 PRINT:INPUT "DO YOU WANT TO TRY ANOTHER ONE   (Y/N)";O$
970 IFLEFT$(O$,1)="N"THENCLS:PRINT@278,"GOOD - BYE, ";NME$;:PRIN
T@406,"NEXT PERSON, PLEASE":GOSUB1100:R=0:T1=0:CLS:GOTO150
992 GOTO200
999 END
1000 PRINT@370,"LEVEL  ";L;:PRINT@434,"S C O R E :";:PRINT@497,R
;"OUT OF";:PRINT@508,T1;
1010 RETURN
1100 S=S+1:NME$=LEFT$(NME$,15):P=INT(R/T1*100):TP=P*L:IFS>10THEN
```

Listing continues

As Easy As...

TRS-80 Disk & Other Mysteries

by H.C. Pennington

This book is the definitive authority on data recovery for the TRS-80 Model I disk system. In almost every case, lost data can be recovered and this book tells you how to do it. From clobbered directories to parity errors, this profusely illustrated data recovery cookbook includes examples and step-by-step instructions for both beginners and professionals.

132 pages. $22.50

Microsoft BASIC Decoded & Other Mysteries

by James Farvour

This book is the only one of its kind. You will learn how the TRS-80 Model I BASIC interpreter works from power-up to power-down. It has the complete disassembled Microsoft BASIC Level II interpreter with over 13,000 lines of detailed comments in one volume. The math, arithmetic and utility routines are fully explained with examples on how to use them. An incredible learning aid for the beginner and a valuable reference for the professional.

312 pages. $29.95

The Custom TRS-80 & Other Mysteries

by Dennis Bathory Kitsz.

A comlete hardware/software how-to-do-it like reverse video, high res graphics, audible keystrokes, 8-track mass storage music synthesizer, and real time clock. Dozens of project hints, tips, and fixes complete with the interface software. If you are a beginner, there are chapters on wire-wrap, soldering and construction methods. If you are a professional, there is a wealth of technical detail.

332 pages. $29.95

BASIC Faster and Better & Other Mysteries

by Lewis Rosenfelder

BASIC is not nearly as slow as you think! This book shows you how to make it faster and better with techniques and tricks that you never dreamed of for Model I, II and III disk systems. You won't find trivial, poorly designed "checkbook" programs here – only useful BASIC techniques and code ready for use in YOUR programs. Tutorial for the beginner, instructive for the advanced, and invaluable for the professional. All routines are available on disk.

290 pages. $29.95

Learn about your TRS-80 computer – it's as easy as 1–2–3–4. These books are easy to read and understand because they are written for you by people just like you. Here are answers and solutions without jargon. There are no end of chapter "tests" – no incomprehensible math formulas – no textbook explanations – just straightforward, plain English.

You will learn how to recover "lost data" on disk, how the BASIC interpreter works, how to make BASIC run faster and better, and how to modify and interface your TRS-80. In other words, you will learn how to make your computer do all the things you want it to do. It's as easy as 1–2–3–4.

Available at computer stores, B. Dalton Booksellers and independent book dealers. **BASIC Faster and Better** is also available at all RADIO SHACK Computer Centers and selected RADIO SHACK stores. (Cat. No. 62-1002) If your dealer is out of stock, order direct. Include $4.00 for shipping and handling. Foreign residents add $11.00 plus purchase price, in U.S. funds.

TRS-80 trademark TANDY Corp.

1953 West 11th Street Upland, CA 91786

Subroutine 6500 is similar but it is used for yes and no questions.

Letting students answer questions this way saves them time and ensures accurate spelling. It also gives them something to play with while they're debating about the right answer.

Other Possibilities

This is a flexible program; many parts can be changed without forcing you to rewrite other parts.

For example, the word lists can be changed or made longer by changing the appropriate data, read, and RND( ) lines. If you want to play with the sentence generator, put a remark at the beginning of line 187 so you can choose any level, and add this line: 519 GOTO 186.

Other types of sentence patterns can be added by changing line 270 and adding more lines similar to lines 350–365. These lines could be added anywhere since they end in a return. Of course, you would also have to change the last question (lines 900–918).

The program can be modified to do other grammar exercises. For example, if you wanted to work on parts of speech, you would need to rewrite lines 315, 365, 415, and 470 because the answers are pulled out of the sentences at this point. The lines that define the questions and answers (520–950) would also have to be changed so subroutine 3000 would ask the right questions and check the correct answers, but the rest of the program can be left intact.

Drills on dependent clauses and subordinating conjunctions would be even easier. I'll let you figure that one out.

If your students get tired of the graphics, you could write your own in subroutines 5000 and 5500. You could even add a graphics game subroutine that could be called from around line 960. Playing time could be based on a certain amount of achievement.

Practical Suggestions

I've found that the optimum time students work with this program is 15 to 30 minutes per group in groups of two or three students. It works well to put high and low-ability students together in the same group. They need at least 15 minutes to start working their way up through the levels, but unless they're really dedicated, they start playing with it after a half hour or so. ■

George Stone can be reached at 5089 Deer Creek Lane, West Bend, WI 53095.

Listing continued

```
S=10
1110 FORN=1TOS:IFP<SP(N)THEN1130
1120 FORY=S-1TONSTEP-1:S$(Y+1)=S$(Y):SP(Y+1)=SP(Y):ST(Y+1)=ST(Y)
:SL(Y+1)=SL(Y):TP(Y+1)=TP(Y):NEXT:S$(N)=NME$:SP(N)=P:ST(N)=T1:SL
(N)=L:TP(N)=TP:GOTO1160
1130 NEXT
1160 PRINT@916,"PRESS <ENTER> TO START";:INPUTO$:RETURN
3000 REM   *** CHECK ANSWER SUB
3010 PRINT:PRINTCHR$(94);Q;"  ";Q$;:GOSUB1000:T1=T1+1:IFLEN(QC$)
>0THENGOSUB6000:PRINTA$:GOTO3030
3020 IFLEN(QYN$)>0GOSUB6500:GOTO3030
3025 A$=INKEY$:IFA$=""THEN3025
3026 IFVAL(A$)<1ORVAL(A$)>4THEN3025ELSEPRINT@64,STRING$(4,153);
3030 IFA$=CA$THENR=R+1:PRINT"RIGHT, ";NME$;", ";:GOTO3060
3035 IFLEN(QC$)=0THEN3050ELSEPRINT"WRONG,  ";QC$;:GOSUB6000
3045 IFA$=CA$THENR=R+.5:PRINT"RIGHT, ";NME$;", ";:GOTO3060
3050 PRINT"WRONG, ";AA$(Q);:GOSUB1000:GOSUB5500:GOTO3065
3060 PRINTAA$(Q):GOSUB1000:GOSUB5000
3065 QC$=""
3070 RETURN
3999 END
4000 REM***  GET READY FOR NEXTQUESTION
4010 CLS
4030 FORN=1TOLS:PRINTCS$(N):PRINT:NEXT:FORN=1TOQ:PRINTN;"  ";AA$
(N):NEXT
4050 Q=Q+1:RETURN
5000 ONRND(6)GOTO5100,5100,5200,5100,5300,5300
5099 REM   ***   PAC MAN EATS
5100 FORN=1TO30:NEXT:X=RND(50):FORN=768+XTO64+XSTEP-64:FORX=1TO4
:PRINT@N-63,F1$(X);:PRINT@N,F2$(X);:PRINT@N+64,B1$;:PRINT@N+128,
B2$;:PRINT@N+192,STRING$(12,128);
5110 FORW=RTO15:IFPEEK(14400)=128THENRETURNELSENEXT:NEXT:NEXT:PR
INT"BURP":FORW=1TO50:NEXT:RETURN
5199 REM   ***   SNAKE   ***
5200 X=510:X1=511:X2=X1:X3=X2:X4=X3:Y1=448:Y=RND(60)+Y1
5210 IFY1=0THENY=-5
5220 I=SGN(Y-X)
5230 IFX<0THEN5240ELSEPRINT@X,"**";
5240 IFX4<0THENRETURNELSEPRINT@X4,"  ";:X4=X2:X2=X
5250 IFX=YORX=Y+1THENX=X-64:Y1=Y1-64:Y=RND(35)+Y1:GOTO5210
5260 X=X+I*2:IFPEEK(14400)=128THENRETURNELSEGOTO5230
5299 REM   ***   WORM   ***
5300 X=4:Y1=4:Y=Y1+RND(56)
5310 FORW=RTO25:IFPEEK(14400)=128THEN5350ELSENEXT:IFX>447THEN535
0ELSEPRINT@X,Z$;:FORW=1TO25:NEXT
5320 IFX=YTHENPRINT@X,U$;:FORW=1TO25:NEXT:X1=X:X=X+64:IFX>447THE
N5350ELSEPRINT@X,U$;:FORW=1TO25:NEXT:Y1=Y1+64:Y=Y1+RND(56):PRINT
@X1,CHR$(200);:GOTO5310
5330 IFX<YTHENPRINT@X,ZR$;:X=X+3:GOTO5310
5340 IFX>YTHENPRINT@X,ZL$;:X=X-2:GOTO5310
5350 PRINT"BYE-BYE":FORW=1TO50:RETURN
5500 REM   ***   PAC-MAN DIES
5505 CLS:FORN=1TO10:X=RND(640):PRINT@X,"  NO WAY, ";NME$;"   ";:
NEXT:PRINT@724,AA$(Q);
5510 N=768:FORX=4TO1STEP-1:PRINT@N-63,F1$(X):PRINT@N,F2$(X):PRIN
T@N+64,B1$:PRINT@N+128,B2$:FORWAIT=1TO100:NEXT:NEXT:PRINT@705,ST
RING$(11,128):FORWAIT=1TO100:NEXT
5520 FORN=1TO150:X=RND(24)-1:Y=RND(6)+38:RESET(X,Y):RESET(X+1,Y)
:IFPEEK(14400)=128THEN5800ELSENEXT
5800 RETURN
6000 REM   ***   MOVE ARROW   ***
6020 Y=65:X=65:PRINT@X,"[";
6030 U=PEEK(14400):IFU=64THENX=X+1:GOTO6040ELSEIFU=1THEN6100ELSE
IFU=32THENX=X-1:GOTO6040ELSEIFU=16THENX=X+128:GOTO6040ELSEGOTO60
30
6040 IFX<64THENX=64ELSEIFX>LS*128-64+LEN(CS$(LS))THENX=64ELSEIFX
=128THENX=192ELSEIFX=191THENX=127ELSEIFX=256THENX=320ELSEIFX=319
THENX=255ELSEIFX>383THENX=64
6060 PRINT@Y," ";:PRINT@X,"[";:Y=X:GOTO6030
6100 X=X+15360-64
6110 X=X-1:U=PEEK(X):IFU=32GOTO6150ELSEGOTO6110
6150 A$=""
6160 X=X+1:U=PEEK(X):IFU=32GOTO6170ELSEA$=A$+CHR$(U):GOTO6160
6170 IFRIGHT$(A$,1)=","ORRIGHT$(A$,1)="."THENA$=LEFT$(A$,LEN(A$)
-1)
6180 PRINTA$;:X=64*(2*LS+Q+2):PRINT@X,"";:RETURN
6500 REM   ***   YES & NO QUESTIONS   ***
6510 A$="":X=690:Y=697:PRINT@624,CHR$(166);:PRINT@688,CHR$(153);
:PRINT@626,QYN$;
6520 U=PEEK(14400):IFU=32THENX=690:Y=697ELSEIFU=64THENX=697:Y=69
0ELSEIFU=1GOTO6550
6530 PRINT@X,"[";:PRINT@Y," ";:GOTO6520
6550 X=X-64+15360
6555 U=PEEK(X):IFU=32THEN6560ELSEA$=A$+CHR$(U):X=X+1:GOTO6555
6560 IFA$="ADJ"THENA$="ADJECTIVE"
6570 QYN$="":PRINT@64*(LS*2+Q+2),"":RETURN
```

✓See List of Advertisers on Page 563

LOAD 80

Avoid the Dangers of Dirty Disks

by Dave Grimes

A clean disk drive runs best, but cleaning a disk drive is easier said than done. This program leads you by the hand through the job.

After reading about the merits of disk system cleanliness, I decided to buy a disk drive head-cleaning kit.

I bought the 3M 7440 kit for 5¼-inch disk drives. This kit includes two 5¼-inch cleaning disks, a four-ounce bottle of cleaning fluid, a spout for the cleaning fluid bottle, and an instruction sheet.

The instructions advise you to dispense cleaning fluid onto the provided disk and to insert the cleaning disk into a drive. You then activate the drive for approximately 30 seconds, remove the disk and write on the label that the disk has been used. After 15 uses, you discard the cleaning disk.

I contemplated this for a while; how was I to turn on a particular disk drive for 30 seconds? For my 40-track drives, how would I know that after 15 uses of the cleaning disk, I had used all 40 tracks equally? The instruction sheet failed to address these questions.

It seems to me that a 40-track disk drive should be able to use the cleaning disk 40 times if I could ensure that a new track were used each time. Also, it would be convenient to turn on a specified disk drive for 30 seconds while the cleaning disk was in that drive.

My program CLEANER/BAS makes all these things possible. It allows you to specify which drive to clean, the track to use on the cleaning disk, and keeps the specified disk-drive motor on for 30 seconds with the read/write head loaded.

How to Use the Program

After entering and verifying the program, run it. The screen shows a banner, gives a brief description of its function, and asks which disk drive you want cleaned. Don't put the cleaning disk in any drive yet, just enter the requested drive number. Next, you're asked which track to use on the cleaning disk. Enter a number from zero to the highest track number on your disk drive, less 1 (zero through 39 for a 40-track drive). When you hit enter the disk drive steps to track zero, then out to the track you entered. The screen clears and a new screen requests that you prepare the cleaning disk for use and insert it in the target drive. Insert the cleaning disk and hit enter, and the target disk drive turns on for 30 seconds. Next, a new screen appears informing you which disk drive you've cleaned, and which track was used on the cleaning disk. You're reminded to remove the cleaning disk so that inadvertently turning on the disk drive won't sweep the read/write head across tracks on the cleaning disk.

The Program

Lines 80–180 of the Program Listing do some housekeeping, and display instructions on the screen. Lines 190, 200, and 210 solicit the disk-drive number (DD) and convert DD from 0, 1, 2, or 3 to 1, 2, 4, or 8 respectively. The converted number is stored in variable DL and is used to select the proper disk drive by POKEing DL into RAM location 14304. Line 220 asks which track number to use on the cleaning disk, and stores the track number in variable CH. Line 230 turns on (selects) the desired drive, and line 240 moves the read/write head to track zero.

Line 270 plugs in the track number that you want to use, and line 290 tells the disk controller to go to that track. Line 310 checks the track register to see if the desired track was located.

After putting the cleaning disk in the targeted drive, as instructed by lines 340 and 350, try to determine that the cleaning disk is actually in the target drive. The simplest way to do this, is to look for write protect in the disk controller status. The cleaning disks don't

A	Response for "Hit enter to continue"
CH	Track to seek on cleaning disk
DD	Input drive #
DL	Delay variable in For...Next loop
DR	Value to POKE to select drive
EF	Error flag. If zero when checked, no error
H1$	Screen heading line
LD	Delay variable in For...Next loop
S	Disk controller status byte
TB	Tab value to center heading on screen

Table 1. List of variables

The Key Box

Model I
32K RAM
Disk Basic
1 Disk Drive
Head-Cleaning Kit

have the notch cut out to allow attempts to write. Subroutine 540–590 makes this check. Subroutine 500–530 checks the status after all commands. If it is determined that a write-protected disk is in the target drive, assume that the disk is the cleaning disk and turn on the drive for 30 seconds with the head loaded against the disk. Lines 380–410 accomplish the 30-second turn on with a delay loop that repeatedly selects the drive. If a drive isn't selected every two seconds or so, the drive shuts off.

This program allows the TRS-80 Model I disk user to control disk head cleaning efforts and keep his disk system clean. ■

Dave Grimes can be reached at 12949 W. Montana Drive, Lakewood, CO 80228. His hobbies include bicycling.

```
10 REM    HEAD CLEANING UTILITY AND CLEANING DISKETTE MANAGER
20 REM                     DECEMBER 1981
30 REM
40 REM                BY DAVE GRIMES
50 REM                  12949 W. MONTANA DR.
60 REM                  LAKEWOOD, CO  80228
70 REM
80 DEFINT A-Z
90 H1$="DISKETTE HEAD CLEANING PROGRAM"
100 TB=(64-(LEN(H1$)))/2  :CLS
110 PRINT TAB(TB) H1$ :PRINT
120 PRINT"THIS  PROGRAM WILL  MANAGE YOUR  HEAD CLEANING DISKETT
ES, BY"
130 PRINT"ALLOWING  YOU  TO SPECIFY  WHICH TRACK  ON THE HEAD CL
EANING"
140 PRINT"DISKETTE YOU WISH TO USE EACH TIME."
150 PRINT"IF YOU  KEEP A RECORD OF WHICH UNUSED TRACK IS NEXT, Y
OU CAN"
160 PRINT"USE  YOUR  CLEANING DISKETTES  35, 40, 77, OR  EVEN 80
 TIMES"
170 PRINT"(DEPENDING ON YOUR DRIVES),  AND USE A CLEAN AREA EACH
 TIME."
180 PRINT
190 INPUT"WHICH DISK DRIVE DO YOU WISH TO CLEAN (0,1,2,3)";DD
200 IF DD <> 0 AND DD <> 1 AND DD <> 2 AND DD <> 3 THEN 190
210 DR=INT(2[DD)              'NOTE EXPONENTIATION
220 INPUT"WHAT IS THE NEXT CLEAN TRACK ON YOUR CLEANING DISKETTE
";CH
230 POKE 14304,DR             'SELECT DRIVE
240 POKE 14316,3              'RESTORE (TRACK 0)
250 GOSUB 500                 'CHECK STATUS
260 POKE 14304,DR             'TURN ON DRIVE MOTOR
270 POKE 14319,CH             'PLUG IN TRACK #
280 GOSUB 500                 'CHECK STATUS
290 POKE 14316,19             'SEEK TRACK
300 GOSUB 500                 'CHECK STATUS
310 IF PEEK(14317) <> CH THEN PRINT"SEEK ERROR ON DRIVE";DD:END
320 CLS :PRINT TAB(TB) H1$ :PRINT
330 PRINT"THE DISK DRIVE TO BE CLEANED WILL BE TURNED ON FOR 30
SECONDS."
340 PRINT:PRINT"PUT THE CLEANING DISKETTE IN DRIVE";DD;"NOW AND
HIT <ENTER>";
350 INPUT A
360 GOSUB 540                 'LOOK FOR WRITE PROTECT
370 IF EF=1 THEN PRINT"*** ERROR ***":GOTO340
380 FOR DL = 1 TO 85
390 POKE 14304,DR
400 FOR LD = 1 TO 100
410 NEXT LD,DL
420 CLS: PRINT TAB(TB) H1$ :PRINT :PRINT
430 PRINT"DRIVE";DD;"CLEANED!":PRINT
440 PRINT"PLEASE MAKE A NOTE THAT TRACK";CH;"OF THE HEAD CLEANIN
G"
450 PRINT"DISKETTE HAS BEEN USED AND SHOULD NOT BE USED AGAIN."
460 PRINT
470 PRINT"YOU SHOULD REMOVE THE HEAD CLEANING DISKETTE NOW."
480 PRINT
490 END
500 S=PEEK(14316)
510 IF S > 127 RETURN
520 IF (S AND 1) <> 0 GOTO 500
530 RETURN
540 POKE 14304,DR             'SELECT DRIVE
550 EF=1                      'SET ERROR FLAG
560 FOR LD = 1 TO 10
570 IF PEEK(14316) > 63 THEN EF=0 :RETURN
580 NEXT LD
590 RETURN
600 END
```

Program Listing

See List of Advertisers on Page 563

Making Labels

by William Nelson

Producing custom labels has never been easier. Written in Basic, this program can easily be modified to work with just about any printer.

The program uses just over 4K memory, and although written for disk, automatically switches to cassette mode if a disk controller is not present in your machine.

The program demonstrates a wide range of Basic routines including screen formatting, string handling, elementary graphics, logic testing, and single-dimensional arrays.

Program Description

When creating a new label, the program asks for height and width of the label, or you can choose several standard label sizes. Print spacing (characters per inch or cpi) is input and the program then draws a proportionally sized label on the screen. The maximum label size is limited by the size of the screen. The program contains many error-trapping routines.

Once the label's dimensions are established, the text lines are whited out and you type in text as with a word processor. If your text exceeds the maximum line length, the line is whited out again and you have to re-enter the text. When all lines are filled, you are asked if they are correct. If not, the editing module is accessed. Then the label can be printed or saved on disk or tape for future use.

The program is menu driven with the following options: create new label, edit label, check printer format, load from disk or tape, save to disk or tape, print label, and end.

The program was written for my Heath H-14 printer with its three print pitches (10, 12, and 16.5 cpi). It is easy to change the escape codes (decimal 27) in lines 710–750 to match your printer.

The program begins by initializing print size to 10 cpi and line spacing to six lines per inch. When using larger labels with more than 10 lines of text, change line 5 by adding the statement DIM T$(n), where n is the anticipated maximum number of lines. Most labels are not that large.

Because of the error-trapping routines, the only error likely to occur is File Not Found during loading from disk. The ON ERROR GOTO statement allows the program to continue even if this happens by returning to the menu. If you suspect other errors are occurring, remove this statement to retrieve Basic error codes.

The menu display and selection is handled by lines 30–60. Normal program exit is at lines 70–90. Each item in the menu is programmed in its own module and these are accessed with the statement in line 60.

Create New Label

Starting at line 100, a new menu is displayed allowing you to choose several standard label sizes or to enter the dimensions of any label desired. The program branches to the Format subroutine (lines 410–460) where variables are input for the number of lines per inch (L) and print pitch (P), which are necessary to calculate the proportions for the graphics label drawn on the screen. There is a branch at this point to the subroutine at line 160 to enter the height (H) and width (W) of the label. From these variables, the program calculates the maximum line length (ML) and the number of lines per label (NL)

Program Listing

```
5 CLEAR 1000:C$=CHR$(176):Z=0:P=10:L=6:N=1:ONERRORGOTO3000
10 S=0:CLS
20 PRINT@25,"LABEL PROCESSOR";:PRINT@91,"Bill Nelson";:PRINT@153
,"February, 1982";
30 PRINT@286,"MENU";:PRINT@407,"1) CREATE NEW LABEL";:PRINT@471,
"2) EDIT LABEL";:PRINT@535,"3) CHECK PRINT FORMAT";:PRINT@599,"4
) LOAD FROM DISK OR TAPE";:PRINT@663,"5) SAVE TO DISK OR TAPE";:
PRINT@727,"6) PRINT LABEL";:PRINT@791,"7) END";
40 PRINT@896,"MAKE SELECTION";:INPUT S
50 IFS<1ORS>7PRINT@896,"SELECTION OUT OF RANGE--TRY AGAIN!!!";:F
ORX=1TO1000:NEXT:PRINT@896,STRING$(36," ");:GOTO40
60 ONSGOTO 100,340,400,500,600,700,70
70 PRINT@896,STRING$(64," ");:PRINT@896,"HAVE YOU SAVED YOUR DAT
A";:INPUTA$:IFLEFT$(A$,1)<>"Y"PRINT@896,STRING$(64," ");:GOTO40
80 PRINT@896,STRING$(64," ");:PRINT@896,"**END OF SESSION**";:FO
R X=1TO2000:NEXT:CLS
90 END
100 S=0:CLS:PRINT@24,"CREATE NEW LABEL";:PRINT@155,"LABEL SIZES"
;:PRINT@215,"1) 1 x 3.5 inches";:PRINT@279,"2) 1.75 x 2.75 inche
s";:PRINT@343,"3) .75 x 2 inches";:PRINT@407,"4) Other";
110 PRINT@576,"MAKE SELECTION";:INPUTS:IFS<1ORS>4PRINT@576,"SELE
```

Listing continues

The Key Box

Level II or Disk Basic
Model III
16K RAM
Disk Drives optional
Printer

Listing continued

```
CTION OUT OF RANGE--TRY AGAIN!!";:FORX=1TO1000:NEXT:PRINT@576,ST
RING$(36," ");:GOTO110
120 IFS=1H=1:W=3.5:GOTO170
130 IFS=2H=1.75:W=2.75:GOTO170
140 IFS=3H=.75:W=2:GOTO170
150 PRINT@704,"ENTER DIMENSIONS OF LABEL";:PRINT@847,"HEIGHT";:I
NPUTH:PRINT@867,"WIDTH";:INPUTW:GOTO170
160 ML=INT(W*P)-1:NL=INT(H*L):XS=62-ML:XF=2*ML+XS+5:IYS=5:YS=IYS
:YF=3+3*(NL+1):ICP=128+(64-ML)/2:CP=ICP:RETURN
170 GOSUB410:IFNL>10ORML>56PRINT@704,STRING$(25," ");:PRINT@847,
STRING$(28," ");:PRINT@704,"LABEL TO LARGE FOR SCREEN--SELECT SM
ALLER SIZE";:FORX=1TO2000:NEXT:PRINT@704,STRING$(46," "):GOTO100

200 CLS:GOSUB 1000
210 FOR J=1 TO NL
220   FOR X=XSTOXF:SET(X,YS+1):SET(X,YS+2):SET(X,YS+3):NEXT
230   T$(J)="":PRINT@CP-4,USING"##";J;:PRINT@CP,C$;
240  A$=INKEY$:IFA$=""THEN240ELSEIFA$=CHR$(8)PRINTA$;:GOSUB2000:
IFBS<1GOTO220ELSEGOTO240ELSET$(J)=T$(J)+A$:IFLEN(T$(J))>ML,T$(J)
="":GOTO220ELSEIFA$<>CHR$(13)PRINTA$;:GOTO240
250   IFZGOTO280
260   YS=YS+3:CP=CP+64
270 NEXT J
280 Z=1:CP=ICP+64*NL
290 PRINT@CP+64,"DO YOU WISH TO CHANGE ANY LINES? (Y/N)";
300 INPUTA$:IFLEFT$(A$,1)<>"Y",340
310 PRINT@CP+56,STRING$(52," ")
320 PRINT@CP+64,"ENTER LINE NUMBER TO CHANGE";:INPUTJ
325 IFJ>NLORJ<1PRINT@CP+128,STRING$(30,32);:PRINT@CP+128,"SELECT
ION OUT OF RANGE--TRY AGAIN!!!";:FORX=1TO1000:NEXT:PRINT@CP+128,
STRING$(36,32);:GOTO320
330 YS=IYS+3*(J-1):CP=ICP+64*(J-1):GOTO220
340 CLS:Z=0
350 CP=ICP
360 FORJ=1TONL
365  PRINT@CP-4,USING"##";J;
370   PRINT@CP," ";T$(J)
380   CP=CP+64
390 NEXT J
395 GOSUB 1000
398 PRINT@CP+64,"DO YOU WISH TO EDIT LABEL (Y/N)";:INPUTA$:IFLEF
T$(A$,1)="Y",Z=1:CP=CP+1:GOTO310ELSEGOTO10
400 GOSUB410:GOTO10
410 CLS:PRINT@29,"FORMAT";:PRINT@158,"CURRENT           NEW";:PRI
NT@261,"PITCH (10,12,16.5 CPI)";:PRINT@325,"LINES PER INCH (6,8)
";:PRINT@389,"NO. LABELS TO BE PRINTED";:PRINT@290,P;:PRINT@302,
CHR$(32);:INPUTP
412 IFP<>10ANDP<>12ANDP<>16.5GOTO410
415 PRINT@302,STRING$(7,32);:PRINT@303,USING"##.#";P
420 PRINT@325,"LINES PER INCH (6,8)";:PRINT@355,L;:PRINT@366,CHR
$(32);:INPUTL
423 IFL<>6ANDL<>8GOTO420
425 PRINT@367,STRING$(5,32);:PRINT@367,L
430 PRINT@389,"NO. LABELS TO BE PRINTED";:PRINT@419,N;:PRINT@430
,CHR$(32);:INPUTN
433 N=INT(N)
434 IFN<1PRINT@389,STRING$(50,32);:PRINT@389,"MUST BE GREATER TH
AN ZERO";:FORX=1TO1000:NEXT:PRINT@389,STRING$(25,32);:GOTO430
435 PRINT@433,STRING$(9,32);:PRINT@430,USING"###";N
440 PRINT@709,"ANY CHANGES (Y/N)";:INPUTA$
450 IFLEFT$(A$,1)<>"N"GOTO410
460 GOSUB160:RETURN
500 T=INP(240):IFT=255GOTO1400
505 CLS:PRINT"LOADING FROM DISK";:PRINT@768,"FILESPEC";:INPUTFS$

510 IFLEN(FS$)>8PRINT@768,STRING$(30,32);:PRINT@768,"FILESPEC TO
O LONG--TRY AGAIN!!";:FORX=1TO1000:NEXT:GOTO500
520 OPEN "I",1,FS$
530 INPUT#1,P,L,H,W,NL,CP,N
540 FORJ=1TONL
550   LINEINPUT#1,T$(J)
560 NEXT
570 CLOSE
580 GOSUB160:GOTO340
600 T=INP(240):IFT=255GOTO1400
605 CLS:PRINT"SAVING TO DISK";:PRINT@768,"FILESPEC ";:INPUTFS$
610 IFLEN(FS$)>10PRINT@768,STRING$(30,32);:PRINT@768,"FILESPEC T
OO LONG--TRY AGAIN!!";:FOR X=1TO1000:NEXT:GOTO600
620 OPEN"O",1,FS$
630 PRINT#1,P;L;H;W;NL;CP;N
640 FOR J=1TONL
650   PRINT#1,T$(J)
660 NEXTJ
670 CLOSE
680 GOTO10
700 ST%=PEEK(14312)AND240:IFST%<>48PRINT@960,"PRINTER NOT READY"
;:FORX=1TO1000:NEXT:GOTO10ELSEGOSUB410
710 IFP=10,D=1ELSEIFP=12,D=20ELSEIFP=16.5,D=40
```

Listing continues

as well as the X and Y axes, starting and finishing points, with the label centered at the top of the screen. The program then returns to the Create module.

The subroutine at lines 1000–1020 provides the graphics algorithm to draw the label outline on the screen. The loop from lines 210–270 prints a white bar across the current line and waits for text to be input via the INKEY statement at line 240. At this point the program checks for the left arrow, CHR$(8), and adjusts the line length count accordingly. To prevent the possibility of backspacing farther than the left margin, the program restarts the entire line when the first space is reached.

Edit Module

When all text lines are filled you have the option to edit any lines. If so, the program jumps back to the appropriate text line by filling in the array subscript for T$(J) and proceeds as before at line 220. The For...Next loop is bypassed at this point through the use of the logical true/false test at line 250. The variable Z is originally set to zero (logical false) for normal loop input. When in the present edit mode, Z is changed to one (logical true) in line 280.

After editing, the finished label is printed on the screen and you have one more chance to change it. You can access the Edit module (lines 310–398) from the main menu at any time. If no further editing is required, the program returns to the menu.

Format Module

The format module sets printer parameters for the Heath H-14 and asks for the number of identical labels to be printed. Modify lines 410–423 for different printers. The Print Using statement is used to provide uniform columns on the display. The Format module is accessed automatically during the Create module as previously explained, but can also be changed prior to printing.

Load and Save Modules

You can save text and format parameters to disk or tape for future use. The Model III's disk controller presence is tested through its input port in lines 500 and 600. If the port's status is 255 decimal, there is no disk controller present and the program branches to the cassette load or save routine.

Model I users must PEEK memory to use this technique, or use only the I/O routine appropriate for your system when typing in the program. Cassette users cannot use commas within the text as they are not accepted by the cassette

Listing continued

```
720 OUT 251,27:OUT 251,117:OUT251,D
730 IF L=6,D=120
740 IF L=8,D=121
750 OUT 251,27:OUT 251,D
760 CLS:PRINT@256,"DO YOU WISH A TEST LABEL";:INPUTA$:IFLEFT$(A$
,1)<>"Y"GOTO820
770 FORJ=1TONL
780     LPRINT"TEST LINE";J:OUT251,10
800 NEXTJ
805 IFH=1.75,OUT251,10:OUT251,10
810 GOTO760
820 FORX=1TON
830  FORJ=1TONL
835 V=LEN(T$(J)):FORQ=1TOV:IFMID$(T$(J),Q,1)=CHR$(13)THENMID$(T$
(J),Q,1)=CHR$(32):NEXT
840      LPRINTT$(J);:OUT 251,10
850   NEXTJ
855 IFH=1.75,OUT251,10:OUT251,10
860 NEXTX
870 GOTO10
1000 FOR X=XSTOXF:SET(X,IYS):SET(X,YF):NEXT
1010 FOR Y=IYSTOYF:SET(XS,Y):SET(XS+1,Y):SET(XF-1,Y):SET(XF,Y):N
EXT
1020 RETURN
1400 CLS:PRINT"LOADING DATA FROM TAPE";:PRINT@768,"READY CASSETT
E PLAYER--PRESS <ENTER> WHEN READY";:INPUTA$
1410 INPUT#-1,P,L,H,W,NL,CP,N
1420 FORJ=1TONL
1430   INPUT#-1,T$(J)
1440 NEXT
1450 GOSUB 160:GOTO340
1500 CLS:PRINT"SAVING DATA ON TAPE";:PRINT@768,"READY CASSETTE P
LAYER--PRESS <ENTER> WHEN READY";:INPUTA$
1510 PRINT#-1,P,L,H,W,NL,CP,N
1520 FOR J=1TONL
1530   PRINT#-1,T$(J)
1540 NEXTJ
1550 GOTO10
2000 BS=LEN(T$(J))-1:IFBS>0THENT$(J)=LEFT$(T$(J),BS):RETURNELSER
ETURN
3000 CLS:PRINT"NO LABEL EXISTS YET";:FOR X=1TO2000:NEXT:GOTO10
```

input routine. Disk users are not bothered by this because of the Disk Basic Line Input statement in line 550.

Printer Routine

The program checks for printer-ready status in line 700. The routine was taken directly from the Model III manual. Escape codes in lines 710–750 change the print size or line spacing in the Heath H-14. Since the printer driver in the Model III is ported rather than memory mapped, I use the port statement OUT 251,n to send these codes as well as line feeds after each line of print. Model I users should use the statement, LPRINT CHR$(n) instead.

You can print a test label to ensure they are correct before multiple printing. To print lines of text, the program at line 835 first searches the text string and changes any carriage returns (decimal 13) to spaces (decimal 32). This way the printer inserts line feeds only at the necessary times. Consequently, it makes no difference whether the printer has an automatic line feed with each carriage return. ■

William Nelson may be reached at 1354 Hackett Street, Beloit, WI 53511.

Circle 29 on Reader Service card.→

Help Protect the Planet From A Wide Variety of Cosmic Crafts That Are Really . . . .
WEIRD!
By Arthur Gleckler
In your quest for survival, battle your way through dozens of different types of strange-looking space vehicles. Watch out. Quick! Put up your shields before they annihilate you!
BIG FIVE SOFTWARE
P.O. Box 9078-185 · Van Nuys, CA 91409 · (213) 782-6861
"Weerd" is now available from your local dealer. If your dealer does not have it, then send us a check or money order for plus for shipping & handling. Outside USA (except Canada) add $6.00 for shipping. California residents add sales tax. Be sure to specify cassette or diskette.
"Weerd" is 100% machine language and includes sound effects playable through the cassette port. Cassette version requires 16K Model 1 or Model 3 TRS-80. Disk version requires 32K Model 1 or Model 3 TRS-80. Disk version features voice sound effects and permanent high scores. Both versions are compatible with the
TRS-80 is a T.M. of Tandy Corp.

Two-Person Space Bomber

by Dale Chermak

Do you tire of single-player games? Do you wish you could challenge a friend to a two-player game?

In Space Bomber, one player defends a planet from bombs dropped by the other. The screen is divided in half by two barriers that move in opposite directions. At the bottom of the screen is the planet's defense base. It is dotted with eight ammo cells and a moving nuclear reactor.

In Space Bomber, one player defends his planet from bombs being dropped by his opponent.

The planet is protected by a defending ship that fires missiles. However, the ship must be stationary while firing, and this makes it an easy target. The ship also defends the ammo cells and the reactor.

The bomber is a constant threat to the planet. It can move at three speeds, and it drops bombs that punch holes in the barriers. The bomber's objective is to destroy either the defending ship, the nuclear reactor, or all of the ammo cells.

Lines	Function
100–440	Initialization
450–500	Data storage area.
510–570	Subroutines to check if defending ship is at edge of screen.
580–1190	Data for base.
1200–1310	Data for bomber.
1320–1350	Data for defending ship.
1360–1470	Data for bomber explosion.
1480–1520	Data for ammo cell explosion.
1530–1570	Data to clear ammo cell after explosion.
1580–1710	Data for messages.
1720–1780	Subroutine to print the data in HL at the screen position in IX.
1790–1840	Subroutine to print barrier.
1850–2080	Subroutine to move the barriers.
2090–2140	Subroutine to print the defending ship.
2150–2490	Subroutine to move the bomber one space right, keep it on the second line, and wrap around if necessary.
2500–2620	Subroutine to move nuclear reactor one space right and wrap around if necessary.
2630–2680	Subroutine to clear screen.
2690–2740	Subroutine to compare HL to DE.
2750–2790	Delay subroutine.
2800–2890	Move barriers and check defending player's keys. Then check for bombing player's keys, move bomber, and do again.
2900–2960	Move defending ship one space right if ship not on edge and print it.
2970–3040	Move defending ship one space left if ship not on edge and print it.
3050–3280	Defending ship fire subroutine.
3290–3450	Print explosion and messages.
3460–3550	See if players want to play again.
3560–3620	Reset ammo count, bomber fire position, and restart.
3630–3760	See if bomber has already fired. If not then check bombing player's keys.
3770–3820	Subroutine to move bomber two spaces.
3830–3990	Check bomber movement keys. If neither hit then move bomber one space. Move nuclear reactor.
4000–4210	Bomber bombing subroutine.
4220–5040	Check bomb position for missile, barrier, ammo cell, nuclear reactor, and defending ship.

Table 1. Program Flow

Game Controls

The bomber is controlled by three keys: The A key moves the bomber two spaces at a time, the S key moves it three spaces at a time, and the D key drops bombs. Bombs can be dropped while the bomber is moving.

The defending ship is also controlled by three keys. Pressing the left arrow moves the defending ship to the left, while using the right arrow moves it to the right. Press enter to fire a missile.

Game Modification

The game can be modified to give handicaps to better players. If you don't want the defending ship's missiles to break through the barrier, delete line 3160. Deleting lines 4230–4270 prevents the bomb from destroying the missile when they are at the same screen position. Also, Model III owners might

The Key Box

Model I
16K RAM
Cassette or Disk Basic
Editor/Assembler

"The bomber is a constant threat to the planet."

want to change the bombs and missiles to characters other than the down and up arrows. The bomb is used in line 4100, and the missile is used in lines 3180, 4040, and 4230.

The Program

Space bomber is written in machine language for fast graphic displays. I use my own subroutines for clearing the screen, delaying, and comparing the HL register to the DE register. You may want to use ROM calls to save memory. Table 1 explains how the program runs. ■

Dale Chermak is in the 11th grade. He lives at 18855 Kinbrace St., Northridge, CA 91326.

Program Listing

```
6400          00100        ORG     6400H
6400 CD1E68   00110 START  CALL    RCLS
6403 DD21D73D 00120        LD      IX,15831
6407 21F566   00130        LD      HL,STMS
640A CD4B67   00140        CALL    PFOS
640D 3E02     00150        LD      A,2
640F F5       00160 DDA    PUSH    AF
6410 010000   00170        LD      BC,0
6413 CD3268   00180        CALL    DELAY
6416 F1       00190        POP     AF
6417 3D       00200        DEC     A
6418 C20F64   00210        JP      NZ,DDA
641B CD1E68   00220        CALL    RCLS
641E DD21C03F 00230        LD      IX,16320
6422 217464   00240        LD      HL,BASE1
6425 CD4B67   00250        CALL    PFOS
6428 110400   00260        LD      DE,4
642B DD19     00270        ADD     IX,DE
642D 219364   00280        LD      HL,BASE2
6430 CD4B67   00290        CALL    PFOS
6433 DD21C03D 00300        LD      IX,15808
6437 CD5867   00310        CALL    FLI
643A DD21003E 00320        LD      IX,15872
643E CD5867   00330        CALL    FLI
6441 219F3F   00340        LD      HL,16287
6444 225C64   00350        LD      (DSP),HL
6447 21403C   00360        LD      HL,15424
644A 225E64   00370        LD      (BSP),HL
644D CDA467   00380        CALL    PDS
6450 08       00390        EX      AF,AF'
6451 D9       00400        EXX
6452 21DE3F   00410        LD      HL,16350
6455 08       00420        EX      AF,AF'
6456 D9       00430        EXX
6457 C33968   00440        JP      INLP
645A 00       00450 FPOL1  DEFB    0
645B 00       00460 FPOL2  DEFB    0
0002          00470 DSP    DEFS    2
0002          00480 BSP    DEFS    2
0002          00490 BFP    DEFS    2
6462 08       00500 ACL    DEFB    8
6463 11BD3F   00510 CRS    LD      DE,16317
6466 CD2C68   00520 CHTD   CALL    HLDE
6469 C0       00530        RET     NZ
646A C34E68   00540        JP      CBMB
646D C9       00550        RET
646E 11803F   00560 CLS    LD      DE,16256
6471 C36664   00570        JP      CHTD
6474 BF       00580 BASE1  DEFB    191
6475 BF       00590        DEFB    191
6476 BF       00600        DEFB    191
6477 BF       00610        DEFB    191
6478 BF       00620        DEFB    191
6479 BF       00630        DEFB    191
647A B0       00640        DEFB    176
647B B0       00650        DEFB    176
647C B0       00660        DEFB    176
647D B7       00670        DEFB    183
647E BB       00680        DEFB    187
647F B0       00690        DEFB    176
6480 B0       00700        DEFB    176
6481 B0       00710        DEFB    176
6482 B7       00720        DEFB    183
6483 BB       00730        DEFB    187
6484 B0       00740        DEFB    176
6485 B0       00750        DEFB    176
6486 B0       00760        DEFB    176
6487 B7       00770        DEFB    183
6488 BB       00780        DEFB    187
6489 B0       00790        DEFB    176
648A B0       00800        DEFB    176
648B B0       00810        DEFB    176
648C B7       00820        DEFB    183
648D BB       00830        DEFB    187
648E B0       00840        DEFB    176
648F B0       00850        DEFB    176
6490 BE       00860        DEFB    190
6491 BF       00870        DEFB    191
6492 00       00880        DEFB    0
6493 BF       00890 BASE2  DEFB    191
6494 BD       00900        DEFB    189
6495 B0       00910        DEFB    176
6496 B0       00920        DEFB    176
6497 B7       00930        DEFB    183
6498 BB       00940        DEFB    187
6499 B0       00950        DEFB    176
649A B0       00960        DEFB    176
649B B0       00970        DEFB    176
649C B7       00980        DEFB    183
649D BB       00990        DEFB    187
649E B0       01000        DEFB    176
649F B0       01010        DEFB    176
64A0 B0       01020        DEFB    176
64A1 B7       01030        DEFB    183
64A2 BB       01040        DEFB    187
64A3 B0       01050        DEFB    176
64A4 B0       01060        DEFB    176
64A5 B0       01070        DEFB    176
64A6 B7       01080        DEFB    183
64A7 BB       01090        DEFB    187
64A8 B0       01100        DEFB    176
64A9 B0       01110        DEFB    176
64AA B0       01120        DEFB    176
64AB BF       01130        DEFB    191
64AC BF       01140        DEFB    191
64AD BF       01150        DEFB    191
64AE BF       01160        DEFB    191
64AF BF       01170        DEFB    191
64B0 BF       01180        DEFB    191
64B1 00       01190        DEFB    0
64B2 B0       01200 BMBR   DEFB    176
64B3 B0       01210        DEFB    176
64B4 9B       01220        DEFB    155
64B5 8C       01230        DEFB    140
64B6 B3       01240        DEFB    179
64B7 BF       01250        DEFB    191
64B8 B3       01260        DEFB    179
64B9 8C       01270        DEFB    140
64BA A7       01280        DEFB    167
64BB B0       01290        DEFB    176
64BC B0       01300        DEFB    176
64BD 00       01310        DEFB    0
64BE B4       01320 DFDS   DEFB    180
64BF 8F       01330        DEFB    143
64C0 B8       01340        DEFB    184
64C1 00       01350        DEFB    0
64C2 B8       01360 EXPB   DEFB    184
64C3 AC       01370        DEFB    172
64C4 89       01380        DEFB    137
64C5 B2       01390        DEFB    178
64C6 9E       01400        DEFB    158
64C7 BA       01410        DEFB    186
64C8 B2       01420        DEFB    178
64C9 9A       01430        DEFB    154
64CA A7       01440        DEFB    167
64CB 8E       01450        DEFB    142
64CC AC       01460        DEFB    172
64CD 00       01470        DEFB    0
64CE BC       01480 EX2    DEFB    188
64CF BF       01490        DEFB    191
64D0 BF       01500        DEFB    191
64D1 BC       01510        DEFB    188
64D2 00       01520        DEFB    0
```

Listing continues

Listing continued

```
64D3 B0          CEX2    DEFB    176
64D4 B0                  DEFB    176
64D5 B0                  DEFB    176
64D6 B0                  DEFB    176
64D7 00                  DEFB    0
64D8 44          BLSM    DEFM    'DEFENDING PLAYER WINS !!!!!
                 PLAY AGAIN <ENTER> --> <--'
6532 00                  DEFB    0
6533 42          DLSM    DEFM    'BOMBING PLAYER WINS !!!!!
                 PLAY AGAIN <ENTER> --> <--'
658D 00                  DEFB    0
658E 57          DSHB    DEFM    'WAY TO GO DEFENDING SHIP. YOU
                 WASTED THE BOMBER !!!!!'
65E5 00                  DEFB    0
65E6 57          DAAC    DEFM    'WAY TO GO BOMBER. YOU WASTED
                 ALL OF THE AMMO CELLS !!!!!!'
6642 00                  DEFB    0
6643 57          YHDS    DEFM    'WAY TO GO BOMBER. YOU WASTED
                 THE DEFENDING SHIP !!!!!'
669B 00                  DEFB    0
669C 57          YBNR    DEFM    'WAY TO GO BOMBER. YOU WASTED
                 THE NUCLEAR REACTOR !!!!'
66F4 00                  DEFB    0
66F5 2A          STMS    DEFM    '** SPACE BOMBER **
               ** BY DALE A. CHERMAK **'
674A 00                  DEFB    0
674B 7E          PFOS    LD      A,(HL)
674C FE00                CP      0
674E C8                  RET     Z
674F DD7700              LD      (IX),A
6752 DD23                INC     IX
6754 23                  INC     HL
6755 C34B67              JP      PFOS
6758 0640        FLI     LD      B,64
675A DD3600BC    FLIA    LD      (IX),188
675E DD23                INC     IX
6760 05                  DEC     B
6761 C25A67              JP      NZ,FLIA
6764 C9                  RET
6765 DD21C03D    USB     LD      IX,15808
6769 DD7E00              LD      A,(IX)
676C 325A64              LD      (FPOL1),A
676F DD7E7F              LD      A,(IX+127)
6772 325B64              LD      (FPOL2),A
6775 0E3F                LD      C,63
6777 DD23        DUSB    INC     IX
6779 DD7E00              LD      A,(IX)
677C DD77FF              LD      (IX+255),A      ;READ +255 AS -1
677F 0D                  DEC     C
6780 C27767              JP      NZ,DUSB
6783 3A5A64              LD      A,(FPOL1)
6786 DD7700              LD      (IX),A
6789 DD21403E            LD      IX,15936
678D 0E3F                LD      C,63
678F DD2B        DUSC    DEC     IX
6791 DD7EFF              LD      A,(IX+255)      ;READ +255 AS -1
6794 DD7700              LD      (IX),A
6797 0D                  DEC     C
6798 C28F67              JP      NZ,DUSC
679B 3A5B64              LD      A,(FPOL2)
679E DD2B                DEC     IX
67A0 DD7700              LD      (IX),A
67A3 C9                  RET
67A4 2A5C64      PDS     LD      HL,(DSP)
67A7 E5                  PUSH    HL
67A8 DDE1                POP     IX
67AA 21BE64              LD      HL,DFDS
67AD CD4B67              CALL    PFOS
67B0 C9                  RET
67B1 2A5E64      MBR     LD      HL,(BSP)
67B4 3620                LD      (HL),32
67B6 23                  INC     HL
67B7 11763C              LD      DE,15478
67BA CD2C68              CALL    HLDE
67BD DAE667              JP      C,PBS
67C0 11803C              LD      DE,15488
67C3 CD2C68              CALL    HLDE
67C6 CAF367              JP      Z,MUOL
67C9 DD21B264            LD      IX,BMBR
67CD DD7E00      DIA     LD      A,(IX)
67D0 FE00                CP      0
67D2 CAFD67              JP      Z,SBP
67D5 77                  LD      (HL),A
67D6 23                  INC     HL
67D7 DD23                INC     IX
67D9 CD2C68              CALL    HLDE
67DC DACD67              JP      C,DIA
67DF 01C0FF              LD      BC,-64
67E2 09                  ADD     HL,BC
67E3 C3CD67              JP      DIA
```

Listing continues

Listing continued

```
67E6 225E64     PBS     LD      (BSP),HL
67E9 E5                 PUSH    HL
67EA DDE1               POP     IX
67EC 21B264             LD      HL,BMBR
67EF CD4B67             CALL    PFOS
67F2 C9                 RET
67F3 01C0FF     MUOL    LD      BC,-64
67F6 09                 ADD     HL,BC
67F7 225E64             LD      (BSP),HL
67FA C3E667             JP      PBS
67FD 2A5E64     SBP     LD      HL,(BSP)
6800 23                 INC     HL
6801 225E64             LD      (BSP),HL
6804 C9                 RET
6805 08         UNR     EX      AF,AF'
6806 D9                 EXX
6807 3620               LD      (HL),32
6809 11E13F             LD      DE,16353
680C CD2C68             CALL    HLDE
680F CA1068             JP      Z,GNRP
6812 23                 INC     HL
6813 368C       PR      LD      (HL),140
6815 08                 EX      AF,AF'
6816 D9                 EXX
6817 C9                 RET
6818 21DE3F     GNRP    LD      HL,16350
681B C31368             JP      PR
681E 21003C     RCLS    LD      HL,15360
6821 3620               LD      (HL),32
6823 11013C             LD      DE,15361
6826 01FF03             LD      BC,1023
6829 EDB0               LDIR
682B C9                 RET
682C 7C         HLDE    LD      A,H
682D BA                 CP      D
682E C0                 RET     NZ
682F 7D                 LD      A,L
6830 BB                 CP      E
6831 C9                 RET
6832 0B         DELAY   DEC     BC
6833 78                 LD      A,B
6834 B1                 OR      C
6835 C23268             JP      NZ,DELAY
6838 C9                 RET
6839 CD6567     INLP    CALL    USB
683C 3A4038             LD      A,(3840H)
683F FE01               CP      1
6841 CA7968             JP      Z,EWH
6844 FE20               CP      32
6846 CA6668             JP      Z,LAH
6849 FE40               CP      64
684B CA5468             JP      Z,RAH
684E CD1B69     CBMB    CALL    OTLP
6851 C33968             JP      INLP
6854 2A5C64     RAH     LD      HL,(DSP)
6857 CD6364             CALL    CRS
685A 3620               LD      (HL),32
685C 23                 INC     HL
685D 225C64     SIT     LD      (DSP),HL
6860 CDA467             CALL    PDS
6863 C34E68             JP      CBMB
6866 2A5C64     LAH     LD      HL,(DSP)
6869 CD6E64             CALL    CLS
686C 110200             LD      DE,2
686F 19                 ADD     HL,DE
6870 3620               LD      (HL),32
6872 11FDFF             LD      DE,-3
6875 19                 ADD     HL,DE
6876 C35D68             JP      SIT
6879 2A5C64     EWH     LD      HL,(DSP)
687C 11C1FF             LD      DE,-63
687F 19                 ADD     HL,DE
6880 E5                 PUSH    HL
6881 FDE1               POP     IY
6883 060D               LD      B,13
6885 FD7E00     LOOP    LD      A,(IY)
6888 FE20               CP      32
688A CA9968             JP      Z,BLNK
688D FEBC               CP      188
688F C2B568             JP      NZ,HITB
6892 FD360020           LD      (IY),32
6896 C34E68             JP      CBMB
6899 FD36005B   BLNK    LD      (IY),91
689D C5                 PUSH    BC
689E CD1B69             CALL    OTLP
68A1 C1                 POP     BC
68A2 FD360020           LD      (IY),32
68A6 11C0FF             LD      DE,-64
68A9 FD19               ADD     IY,DE
68AB 05                 DEC     B
```

Listing continues

See List of Advertisers on Page 563

Listing continued

```
68AC CA3968   03260        JP    Z,INLP
68AF CD6567   03270        CALL  USB
68B2 C38568   03280        JP    LOOP
68B5 2A5E64   03290 HITB   LD    HL,(BSP)
68B8 E5       03300        PUSH  HL
68B9 DDE1     03310        POP   IX
68BB 21C264   03320        LD    HL,EXPB
68BE CD4B67   03330        CALL  PFOS
68C1 010000   03340        LD    BC,0
68C4 CD3268   03350        CALL  DELAY
68C7 CD1E68   03360        CALL  RCLS
68CA 218E65   03370        LD    HL,DSHB
68CD DD21533D 03380        LD    IX,15699
68D1 CD4B67   03390        CALL  PFOS
68D4 010000   03400        LD    BC,0
68D7 CD3268   03410        CALL  DELAY
68DA 21D864   03420        LD    HL,BLSM
68DD DD21D33D 03430 PMAC   LD    IX,15827
68E1 CD4B67   03440        CALL  PFOS
68E4 DD21293E 03450        LD    IX,15913
68E8 DD360058 03460 KLP    LD    (IX),88
68EC 3A4038   03470        LD    A,(3840H)
68EF FE01     03480        CP    1
68F1 CA0769   03490        JP    Z,FXV
68F4 01C409   03500        LD    BC,2500
68F7 CD3268   03510        CALL  DELAY
68FA DD36002A 03520        LD    (IX),42
68FE 01C409   03530        LD    BC,2500
6901 CD3268   03540        CALL  DELAY
6904 C3E868   03550        JP    KLP
6907 3E08     03560 FXV    LD    A,8
6909 326264   03570        LD    (ACL),A
690C 210000   03580        LD    HL,0
690F 226064   03590        LD    (BFP),HL
6912 010000   03600        LD    BC,0
6915 CD3268   03610        CALL  DELAY
6918 C30064   03620        JP    START
691B 2A6064   03630 OTLP   LD    HL,(BFP)
691E 7D       03648        LD    A,L
691F FE00     03650        CP    0
6921 C2A469   83660        JP    NZ,BHF
6924 3A0138   03670        LD    A,(3801H)
6927 FE10     03680        CP    16
6929 CA8069   03690        JP    Z,DKH
692C FE12     03700        CP    18
692E CA8069   03710        JP    Z,DKH
6931 3A0138   03720 OTLP1  LD    A,(3801H)
6934 FE02     03730        CP    2
6936 CA3E69   03740        JP    Z,MB2
6939 FE12     03750        CP    18
693B C25069   03768        JP    NZ,NEX
693E CDB167   03770 MB2    CALL  MBR
6941 01C409   03780        LD    BC,2500
6944 CD3268   03790        CALL  DELAY
6947 CDB167   03800        CALL  MBR
694A 01C409   03810        LD    BC,2580
694D C37969   03820        JP    JBTI
6950 3A0438   03830 NEX    LD    A,(3804H)
6953 FE08     03840        CP    8
6955 C27369   03850        JP    NZ,NEX1
6958 CDB167   03860        CALL  MBR
695B 017206   03870        LD    BC,1650
695E CD3268   03880        CALL  DELAY
6961 CDB167   03890        CALL  MBR
6964 017206   03900        LD    BC,1650
6967 CD3268   03910        CALL  DELAY
696A CDB167   03920        CALL  MBR
696D 017206   03930        LD    BC,1650
6970 C37969   03940        JP    JBTI
6973 CDB167   03950 NEX1   CALL  MBR
6976 018813   03960        LD    BC,5008
6979 CD3268   03970 JBTI   CALL  DELAY
697C CD0568   03980        CALL  DNR
697F C9       03990        RET
6980 2A5E64   04000 DKH    LD    HL,(BSP)
6983 114500   04010        LD    DE,69
6986 19       04020        ADD   HL,DE
6987 7E       04030        LD    A,(HL)
6988 FE5B     04040        CP    91
698A C29769   04050        JP    NZ,PYB
698D FD360020 04060 YHEM   LD    (IY),32
6991 CD3169   04070        CALL  OTLP1
6994 C33968   04080        JP    INLP
6997 226064   04090 PYB    LD    (BFP),HL
699A 365C     04100        LD    (HL),92
699C E5       04110        PUSH  HL
699D CD3169   04120        CALL  OTLP1
69A0 E1       84130        POP   HL
69A1 3620     04140        LD    (HL),32
```

Listing continues

Listing continued

```
69A3 C9       04150       RET
69A4 2A6064   04160 BHF   LD      HL,(BFP)
69A7 114000   04170       LD      DE,64
69AA 19       04180       ADD     HL,DE
69AB 11FF3F   04190       LD      DE,16383
69AE CD2C68   04200       CALL    HLDE
69B1 D2CF69   04210       JP      NC,CFAR
69B4 7E       04220       LD      A,(HL)
69B5 FE5B     04230       CP      91
69B7 C2C369   04240       JP      NZ,OKKT
69BA 210000   04250       LD      HL,0
69BD 226064   04260       LD      (BFP),HL
69C0 C38D69   04270       JP      YHEM
69C3 FE20     04280 OKKT  CP      32
69C5 CA9769   04290       JP      Z,PYB
69C8 FEBC     04300       CP      188
69CA C2D869   04310       JP      NZ,OKKT1
69CD 3620     04320       LD      (HL),32
69CF 210000   04330 CFAR  LD      HL,0
69D2 226064   04340       LD      (BFP),HL
69D5 C33169   04350       JP      OTLP1
69D8 FEB7     04360 OKKT1 CP      183
69DA C2076A   04370       JP      NZ,OKKT2
69DD 2B       04380 PEX2  DEC     HL
69DE E5       04390       PUSH    HL
69DF DDE1     04400       POP     IX
69E1 21CE64   04410       LD      HL,EX2
69E4 CD4867   04420       CALL    PFOS
69E7 01C409   04430       LD      BC,2500
69EA CD3268   04440       CALL    DELAY
69ED 11FCFF   04450       LD      DE,-4
69F0 DD19     04460       ADD     IX,DE
69F2 21D364   04470       LD      HL,CEX2
69F5 CD4867   04480       CALL    PFOS
69F8 3A6264   04490       LD      A,(ACL)
69FB FE01     04500       CP      1
69FD CA106A   04510       JP      Z,OACL
6A00 3D       04520       DEC     A
6A01 326264   04530       LD      (ACL),A
6A04 C3CF69   04540       JP      CFAR
6A07 FEBB     04550 OKKT2 CP      187
6A09 C22F6A   04560       JP      NZ,OKKT3
6A0C 2B       04570       DEC     HL
6A0D C3DD69   04580       JP      PEX2
6A10 010000   04590 OACL  LD      BC,0
6A13 CD3268   04600       CALL    DELAY
6A16 CD1E68   04610       CALL    RCLS
6A19 21E665   04620       LD      HL,DAAC
6A1C DD21533D 04630 PHAA  LD      IX,15699
6A20 CD4B67   04640       CALL    PFOS
6A23 010000   04650       LD      BC,0
6A26 CD3268   04660       CALL    DELAY
6A29 213365   04670       LD      HL,DLSH
6A2C C3DD68   04680       JP      PMAC
6A2F FE8C     04690 OKKT3 CP      140
6A31 C2506A   04700       JP      NZ,OKKT4
6A34 21C03D   04710       LD      HL,15808
6A37 36BF     04720       LD      (HL),191
6A39 11C13D   04730       LD      DE,15809
6A3C 013F02   04740       LD      BC,575
6A3F EDB0     04750       LDIR
6A41 018813   04760       LD      BC,5000
6A44 CD3268   04770       CALL    DELAY
6A47 CD1E68   04780       CALL    RCLS
6A4A 219C66   04790       LD      HL,YHNR
6A4D C31C6A   04800       JP      PHAA
6A50 FE8F     04810 OKKT4 CP      143
6A52 C2886A   04820       JP      NZ,OKKT5
6A55 2A5C64   04830 PHAB  LD      HL,(DSP)
6A58 E5       04840       PUSH    HL
6A59 DDE1     04850       POP     IX
6A5B DD3600BF 04860       LD      (IX),191
6A5F DD3601BF 04870       LD      (IX+1),191
6A63 DD3602BF 04880       LD      (IX+2),191
6A67 018813   04890       LD      BC,5000
6A6A CD3268   04900       CALL    DELAY
6A6D DD360020 04910       LD      (IX),32
6A71 DD360120 04920       LD      (IX+1),32
6A75 DD360220 04930       LD      (IX+2),32
6A79 010000   04940       LD      BC,0
6A7C CD3268   04950       CALL    DELAY
6A7F CD1E68   04960       CALL    RCLS
6A82 214366   04970       LD      HL,YHDS
6A85 C31C6A   04980       JP      PHAA
6A88 FEB4     04990 OKKT5 CP      180
6A8A CA556A   05000       JP      Z,PMAB
6A8D FEB8     05010       CP      184
6A8F CA556A   05020       JP      Z,PMAB
6A92 C3CF69   05030       JP      CFAR
6400          05040       END     START
00000 TOTAL ERRORS
```


✓See List of Advertisers on Page 563

Don't miss out! Take advantage of these great bargains during

ACCESS UNLIMITED'S

SAVE $50.00 on fully-tested PERCOM Disk Drive for Model III*
Reg. $499.00. Now $449.00

Percom's internally-mounted drives, with their widely-acclaimed disk controllers, are completely pre-tested with a 48 hour Burn-in. So, you know it works right when you get it. Choose single or double-density storage capacity.

A First-Drive System includes the four-drive disk controller, your choice of one or two drives, power supplies, cables and a fully-illustrated, easy-to-understand owner's manual.

Top-of-the-line Access Unlimited addons for the Model III* Disk Drive only $334.00 Single, and $628.00 Double.

Access Unlimited's drives for the Model III* bring you all the outstanding features of the Percom Drive, plus the superb, deluxe quality of the Access Unlimited workmanship that goes into the precision-made equipment which bears our name.

Make sure it's done right. Let us install your First-Drive System.

Purchase your Model III* Drives from us and for $39.95 plus shipping, you can have our experienced professionals perform a 48-hour burn-in of your Model III* computer, install the drive system, and check out your expanded system to make sure everything works correctly. Call for pricing of complete Model III* system with 2 to 4 Access Unlimited or Percom drives.

Get the best! Our own fully-tested drives for as low as $275.00

Every Access Unlimited Floppy and Flippy Disk Drive is electrically and mechanically tested, then burned-in for 48 hours under operating conditions. The signed test list in your shipping carton is proof of that.

Our floppy disk drives offer you either single or double density. With double-density, it stores up to 364 Kbytes, depending on the format. And you can use both sides of the diskette on our flippy drives.

SATISFACTION GUARANTEED! If you're not completely satisfied with your Access Unlimited Disk Drive, return it within 15 days for a full refund. All our drives have a comprehensive 90-day limited warranty.

— Sale ends Feb. 28, 1983 —

(But some prices will be good after Sale ends. Please call 1-800-527-3475 for pricing.)

Save $$ on our most popular items!

Percom Data Separator (reg. $29.95)	now $23.95
Screens for Models I*, II*, III Green, Lt. Blue, Dark blue, and Amber. Bronze for color video (reg. $24.95)	now $14.95
Head Cleaning Kit (reg. $29.95)	now $19.95
Drive Numbering Tabs, pkg. 0-3 (reg. $4.50)	now $3.95
Flip-N-File 5¼" (reg. $39.95)	now $24.95
Flip-N-File 8" (reg. $54.95)	now $39.95

AFD 100-40-track single-density drives $275.00

JUST PURCHASED! Overstock of Japanese-manufactured absolutely top quality CP/M^{X} based business computers. From 160 KB to 40 + MB.

We have just added approximately $100,000.00 worth of brand new business and game software.

Call or write for our new FREE catalog.

Electric Crayon™ Close-out—only 80 left. When these are sold, there will be no more! reg. $289.00 now $149.95

Use the electric crayon with your own monitor or color TV set. Use it with your Model I or III TRS-80™ and a color monitor or color TV set for a very low cost, high quality color computer. (Hook up instructions furnished.)

XTRA Special—Soft sector diskettes—Carload purchase—As low as $19.90 for box of 10

SSSD, with hub, 5¼"	$19.90 box of 10
SSDD, with hub, 5¼"	22.80 box of 10
DSSD, with hub, 5¼"	29.90 box of 10
DSDD, with hub, 5¼"	33.50 box of 10
SSSD, 8"	22.75 box of 10
SSSD, 8", high performance	22.90 box of 10
DSDD 8"	31.95 box of 10

One year limited warranty against manufacturing defects

✓26

SPECIALLY TARGETED. Works in harmony with your TRS80 Model I or III. No more floppy shuffle. Fast one pass compiler that doesn't generate unnecessary intermediate files which require further assembly. Compatible with TRSDOS, LDOS, DOSPLUS, NEWDOS.

PROFESSIONALLY DESIGNED. Crafted by a project team with over 15 years of Pascal programming experience. A powerful Pascal that includes all the features necessary for professional program development including random files and dynamic strings.

INCREDIBLE SPEED. Resultant stand alone programs execute 10-20 times faster than the equivalent Basic or up to 4-1/2 times faster than U.C.S.D. programs.

POWERFUL TEXT EDITOR. Has full screen capability for easy program or document preparation.

EXCEPTIONAL DOCUMENTATION. Includes 250 page documentation pkg. that guides you step by step.

OPTIONAL ADVANCED DEVELOPMENT PKG. Includes Optimizer and Native Code Generator for even faster executing programs. ADP: $125 + shipping.

ALCOR PASCAL: A new standard of excellence from the professionals at Alcor Systems.

CALL OR WRITE for a free 20 page technical brochure with all the facts. 214-226-4476. Pascal System: $199 + shipping.

ALCOR SYSTEMS

Alcor Systems
800 W. Garland Ave.
Garland, Texas 75040

~75

	TRADEMARK
TRS80	Tandy Corporation
UCSD	University Calif. San Diego
Alcor Pascal	Alcor Systems

Dealer Inquiries Invited

Dual-Voice Music Synthesizer

by Lee Morgenstern

Those of you who are bored with your computer playing only one note at a time can learn how to get multiple notes through software.

Are you tired of hearing only single-note-at-a-time tunes from your TRS-80? This program demonstrates how to create musical pieces that play through the cassette port and have both treble and bass voices. Best of all, it requires no extra hardware.

Some musical pieces require two voices. Previously, hardware music synthesizers, which are expensive, were the only way to produce multiple voices. Now, however, it is possible to produce them completely with software.

Theory of Operation

Other programs produce music by generating square waves with two amplitudes, or voltage levels, and causing the cassette output to vibrate. This program creates a more advanced wave by controlling the three voltage levels available at the cassette port. With three levels, it produces a wave resulting from the amplitude addition of two separate square waves as in Fig. 1. When the program sends this wave through the cassette port, the output sounds like two independent tones played at the same time.

Programming amplitude addition is not as tricky as you might think. The cassette commands have an interesting pattern. Table 1 shows the binary command values that produce the three voltage levels at the cassette auxiliary output.

Bit 2 turns on the cassette motor and bits 0 and 1 control the voltage. Complementing either bit 0 or 1 produces a voltage change of one level by "vibrating" bit 1 to produce the treble notes and independently doing the same to bit 0 to produce the bass notes.

Program Features

The Dual-Voice Music Synthesizer is written in Basic and Assembly language. The Basic part creates and edits a list of treble and bass notes, and an Assembly-language USR subroutine plays the music. The program uses

command bit:	2 1 0		
	1 1 0	=	0.00 volts
	1 1 1	=	0.46 volts
	1 0 0	=	0.46 volts
	1 0 1	=	0.85 volts

Table 1. Cassette-port Command Values for the Voltage Levels

Variable	Meaning
M$	USR machine code
N$	names of the notes
N	frequency counts of the notes
T()	treble note list
B()	bass note list
D()	dual-voice play buffer

Table 2. Important Basic Variables

Byte	Contents
1	note duration MSB
2	note duration LSB
3	treble note frequency count
4	treble note octave count
5	bass note frequency count
6	bass note octave count
7	treble note rest flag—2 if played, 0 if rest
8	bass note rest flag—1 if played, 0 if rest

Table 3. Play Buffer Format

The Key Box

Model I or III
16K, 32K RAM
Cassette or Disk Basic
Cassette Port Amplifier
Printer Optional

POKE statements to enter the USR machine code into a string.

The musical capabilities of the program are a 5-octave note range, whole note to 64th note timing, sharps, and rests. In a 16K TRS-80, there is room for 700 notes for treble and bass each. In a 48K machine, this expands to 3,400 notes each.

The music plays through an amplifier and speaker, or records on cassette tape, while a special feature changes the timing of the piece.

The file handling provides loading and saving of music files using either cassette or disk. The editing features consist of append, insert, delete, display, and print functions for both treble and bass note lists.

Entering the Program

Type Program Listing 1 exactly as it appears and it will work for Level II Basic or Disk Basic. You may, however, remove certain lines depending on your system.

For Level II Basic using a cassette, you may omit lines 1650–1800, 1880–2030, 3610–3650, and 3670–3710.

For Disk Basic using disks for file storage, you may omit lines 1650–1720, 1800–1840, 1880–1950, 2030–2070, 3610–3650, 3670, and 3720–3750.

For Disk Basic using cassette for file storage, you may omit lines 1650–1790, 1880–2020, 3610–3650, 3670, and 3720–3750.

Operating the Program

A typical procedure for creating and playing music consists of the following steps:

- Enter the treble and bass notes using the editing functions.
- Save the note lists on disk or tape.
- Build the play buffer.
- Play the music.
- Change the timing if necessary.

Two menus contain all the program

```
+--+  +--+  +--+  +--+
+  +--+  +--+  +--+  +--      a.  treble note


+-----+     +-----+
+     +-----+     +-----      b.  bass note


+--+        +--+
|  +-----+  |  +-----+
+        +--+        +--      c.  treble and bass combined
```

Fig. 1. Two voices are created by adding the amplitudes of two waves. The two waves (1a and 1b), each having two amplitude levels, combine to form wave 1c, having three amplitude levels.

```
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
|  | octave |  note   |        duration          |
|  |  1-5   |  0 - 12 |         1 - 64           |
```

Fig. 2. Packing of Treble and Bass Notes

Program Listing 1

```
1000 'DUAL-VOICE MUSIC SYNTHESIZER
1010 'BY LEE MORGENSTERN, MAY, 1982
1020 '
1030 CLS: CLEAR 256: DEFINT A-Z
1040 PRINT"DUAL-VOICE MUSIC SYNTHESIZER"
1050 PRINT: PRINT"READING DATA:"
1060 E1$="TREBLE OR BASE LIST EMPTY"
1070 E2$="PLAY BUFFER NOT BUILT"
1080 E3$="NOTE LIST EMPTY"
1090 E4$="IMPROPER NOTE"
1100 E5$="IMPROPER OCTAVE"
1110 M$=STRING$(152,0)
1120 P!=PEEK(VARPTR(M$)+1) + 256*PEEK(VARPTR(M$)+2)
1130 IF P!>32767 THEN P!=P!-65536
1140 P=P!
1150 FOR X=1 TO 152: PRINT@141,152-X;
1160 READ Q: POKE P,Q: P=P+1: NEXT
1170 DATA 205,127, 10,243,235,221, 33,  0,  0,221
1180 DATA  25,221, 78,  8,221, 78,  1,221, 35,221
1190 DATA  35, 17,  0,  0,217, 62,  4,  0,221,102
1200 DATA   0,221,110,  1,221, 70,  2,221, 70,  3
1210 DATA 221, 94,  4,221, 86,  5, 62,  6, 60,185
1220 DATA  48, 72,187, 56, 20,121,147, 79,125,147
1230 DATA 111,221, 94,  4, 62,  2, 21, 32, 14,  0
1240 DATA 221,174,  7,211,255,  8,221, 86,  5, 62
1250 DATA   5, 24,  4, 60,185, 48, 37,189, 56,214
1260 DATA 121,149, 79,123,149, 95,221,110,  1, 62
1270 DATA   2, 37, 32,200,217,221, 25, 11,120,177
1280 DATA 200,217,  6,  1, 22,  1,221,102,  0,221
1290 DATA 110,  1, 24,170,123,145, 95,125,145,111
1300 DATA 221, 78,  2, 62,  3,  5, 32,170,  8,221
1310 DATA 174,  6,211,255,  0,221, 70,  3, 62,  5
```

Listing 1 continues

Listing 1 continued

```
1320 DATA  24,156
1330 DIM N$(12),N(12)
1340 FOR F=0 TO 12: READ N$(F),N(F): NEXT
1350 DATA R,255,B,50,A#,53,A,56,G#,59,G,63,F#,67,F,71,E,75
1360 DATA D#,79,D,84,C#,89,C,94
1370 PRINT:PRINT"ALLOCATING MEMORY"
1380 M=MEM/12-100
1390 DIM T(M),B(M),D(4*M)
1400 BF=0: T(0)=0: B(0)=0: TD=0: BD=0: TE=0: TF=0: BE=0: BF=0
1410 F$="": EM=0: L=0: D=0: E=0: N=0: T=0: B=0: M1=0
1420 C1=255: C2=256: C3=3840: C4=192: C5=4096
1430 '
1440 '**** MAIN MENU
1450 CLS
1460 PRINT"DUAL-VOICE MUSIC MAIN MENU"
1470 PRINT
1480 PRINT"<1> LOAD MUSIC FILE"
1490 PRINT"<2> SAVE MUSIC FILE"
1500 PRINT"<3> EDIT MUSIC"
1510 PRINT"<4> BUILD PLAY BUFFER"
1520 PRINT"<5> CHANGE TIMING"
1530 PRINT"<6> PLAY MUSIC"
1540 PRINT
1550 M=0: INPUT"SELECTION";M
1560 IF M<1 OR M>6 THEN 1550
1570 IF M=4 AND (T(0)=0 OR B(0)=0) THEN PRINT E1$: GOTO 1550
1580 IF M>=5 AND BF=0 THEN PRINT E2$: GOTO 1550
1590 ON M GOSUB 1630,1870,2100,3190,3520,3600
1600 GOTO 1450
1610 '
1620 '**** LOAD MUSIC FILE
1630 CLS
1640 BF=0
1650 PRINT"<1> LOAD FROM DISK"
1660 PRINT"<2> LOAD FROM CASSETTE"
1670 PRINT"<3> LOAD FROM CASSETTE WITH DISK BASIC"
1680 PRINT
1690 M=0: INPUT"SELECTION";M
1700 IF M<1 OR M>3 THEN 1690
1710 ON M GOSUB 1730,1810,1800
1720 RETURN
1730 INPUT"MUSIC FILE SPEC";F$
1740 OPEN"I",1,F$
1750 INPUT#1,T(0),B(0)
1760 IF T(0)>0 THEN FOR X=1 TO T(0): INPUT#1,T(X): NEXT
1770 IF B(0)>0 THEN FOR X=1 TO B(0): INPUT#1,B(X): NEXT
1780 CLOSE
1790 RETURN
1800 CMD"T"
1810 INPUT#-1,T(0),B(0)
1820 IF T(0)>0 THEN FOR X=1 TO T(0): INPUT#-1,T(X): NEXT
1830 IF B(0)>0 THEN FOR X=1 TO B(0): INPUT#-1,B(X): NEXT
1840 RETURN
1850 '
1860 '**** SAVE MUSIC FILE
1870 CLS
1880 PRINT"<1> SAVE TO DISK"
1890 PRINT"<2> SAVE TO CASSETTE"
1900 PRINT"<3> SAVE TO CASSETTE WITH DISK BASIC"
1910 PRINT
1920 M=0:INPUT"SELECTION";M
1930 IF M<1 OR M>3 THEN 1920
1940 ON M GOSUB 1960,2040,2030
1950 RETURN
1960 INPUT"MUSIC FILE SPEC";F$
1970 OPEN"O",1,F$
1980 PRINT#1,T(0),B(0)
1990 IF T(0)>0 THEN FOR X=1 TO T(0): PRINT#1,T(X): NEXT
2000 IF B(0)>0 THEN FOR X=1 TO B(0): PRINT#1,B(X): NEXT
2010 CLOSE
2020 RETURN
2030 CMD"T"
2040 PRINT#-1,T(0),B(0)
2050 IF T(0)>0 THEN FOR X=1 TO T(0): PRINT#-1,T(X): NEXT
2060 IF B(0)>0 THEN FOR X=1 TO B(0): PRINT#-1,B(X): NEXT
2070 RETURN
2080 '
2090 '**** EDIT MUSIC
2100 EM=0
2110 CLS: PRINT"DUAL-VOICE MUSIC EDITOR"
2120 IF EM=0 THEN PRINT"TREBLE:";T(0); ELSE PRINT"BASE:";B(0);
2130 PRINT"NOTES"
2140 PRINT
2150 PRINT"<0> TOGGLE TREBLE/BASE"
2160 PRINT"<1> APPEND TO END OF LIST"
2170 PRINT"<2> INSERT NOTES"
2180 PRINT"<3> DELETE NOTES"
```

Listing 1 continues

options—the main menu and the edit menu. The main menu has six options:

Dual-Voice Music Main Menu
(1) Load Music File
(2) Save Music File
(3) Edit Music
(4) Build Play Buffer
(5) Change Timing
(6) Play Music

Option 1 loads the treble and bass lists of a previously created music file from either cassette or disk to memory. Option 2 saves the treble and bass lists currently in memory to cassette or disk. Option 3 calls the edit menu to create or change music in memory. Option 4 builds a play buffer from the treble and bass lists in memory. Option 5 changes the timing of the play buffer. Option 6 calls the USR subroutine to play the music.

Option 3 of the main menu calls the edit menu:

Dual-Voice Music Editor
Treble: XXX Notes
(0) Toggle Treble/Bass
(1) Append to End of List
(2) Insert Notes
(3) Delete Notes
(4) Display Notes
(5) Print Notes
(6) Exit

The second line of the edit menu displays the current number of notes in the treble or bass list. Option 0 switches back and forth between the two lists. Options 1-5 act on the list currently selected. Option 1 adds notes to the end of a list or starts a new list. Option 2 inserts notes at the start or in the middle of an existing list. Option 3 deletes one note or a block of notes. Option 4 displays the note list on the screen. Option 5 prints the note list on a line printer, formatted in multiple columns. Option 6 exits back to the main menu.

Creating the Note Lists

When appending or inserting notes, follow the format of duration, octave-note as in 4,5F#. This means quarter-note duration, 5th octave and F sharp. Durations range from 1, for a whole note, to 64, for a 64th note. Octaves range from 1, for the lowest pitch, to 5, for the highest pitch. Notes are A–G with an optional #. To enter a rest, type the duration and then R as in 8,R. This means an 8th note rest. The range of notes is from 1C to 5B. To terminate the list, enter 0,0.

Listing 1 continued

```
2190 PRINT"<4> DISPLAY NOTES"
2200 PRINT"<5> PRINT NOTES"
2210 PRINT"<6> EXIT"
2220 PRINT
2230 M=-1: INPUT"SELECTION";M
2240 IF M=6 THEN RETURN
2250 IF M=0 THEN EM=1-EM: GOTO 2110
2260 IF M<1 OR M>5 THEN 2230
2270 IF EM=0 THEN L=T(0) ELSE L=B(0)
2280 IF M>=2 AND M<=6 AND L=0 THEN PRINT E3$: GOTO 2230
2290 ON M GOSUB 2330,2540,2750,2860,3000
2300 GOTO 2110
2310 '
2320 '**** APPEND NOTES
2330 CLS
2340 PRINT"APPENDING ";
2350 IF EM=0 THEN PRINT"TREBLE"; ELSE PRINT"BASE";
2360 PRINT" NOTES"
2370 PRINT
2380 PRINT"DURATION,NOTE"
2390 L=L+1
2400 PRINT USING "####:";L;
2410 D=-1: INPUT D,F$: IF D=0 THEN RETURN
2420 IF F$="R" THEN E=0: F=0: GOTO 2480
2430 IF LEN(F$)<2 THEN PRINT E4$: GOTO 2400
2440 E=VAL(LEFT$(F$,1)): IF E<1 OR E>5 THEN PRINT E5$: GOTO2400
2450 F=1: F$=MID$(F$,2)
2460 IF F$<>N$(F) THEN F=F+1: IF F<13 THEN 2460
2470 IF F>12 THEN PRINT E4$: GOTO 2400
2480 IF EM=0 THEN T(0)=L ELSE B(0)=L
2490 P=4096*E + 256*F + D
2500 IF EM=0 THEN T(L)=P ELSE B(L)=P
2510 BF=0: GOTO 2390
2520 '
2530 '**** INSERT NOTES
2540 CLS
2550 N=0: INPUT"INSERT BEFORE WHICH NOTE";N
2560 IF N<1 OR N>L THEN 2550
2570 PRINT"DURATION,NOTE"
2580 PRINT USING "####:";N;
2590 D=-1: INPUT D,F$: IF D=0 THEN RETURN
```

Listing 1 continues

The editor numbers all the notes. Use these numbers to refer to the notes when deleting or inserting. Enter the starting and ending note numbers to delete a series of notes. The starting and ending numbers would be the same number if you wanted to delete only one note. To insert notes, enter the number following the insert point; this allows insertion before the first note. Use the append option to insert notes at the end of the list.

To replace one or more notes, first delete them and then insert the replacements. The editor controls the numbers so that the first note number you delete becomes the note number to insert. For example, if you want to replace notes 33–40, select the delete option and enter 33,40. Then select the insert option and enter 33 as the insert point. Type the replacement notes and then 0,0.

The display option lists the notes on the screen in a single column. To freeze the list as it scrolls by, press shift, @. Press any key to resume scrolling.

If you have a printer connected to the parallel printer port, select the print option to produce a multiple-column listing of the notes. Each column is 16 characters wide. This permits five columns on an 80-character-per-line print-

er or eight columns on a 132-character-per-line printer.

Playing the Music

Before playing the piece, you must select the build-play-buffer option. This option creates a combined buffer from the treble or bass lists so the USR subroutine can efficiently create the dual-voice melodies. Once you have built the play buffer, you can play the music repeatedly without delay.

The musical piece can be played in two ways. The best way is to connect an amplifier and speaker to the auxiliary output of the cassette port. The extra hardware is not necessary for the second method. Just connect your cassette recorder in the standard way, set the cassette to record, select the play-music option, wait for the main menu to reappear on the screen, stop the recorder, rewind to the start point, unplug the cables, and press the cassette play button.

If the piece plays too fast or too slow, select the change-timing option. Enter a multiplier for the duration. Enter 2 if you want the piece to play slower, or enter 0.5 if you want the piece to play faster. The multiplier acts only on the play buffer, so the original treble and

Listing 1 continued

```
2600 IF F$="R" THEN E=0: F=0: GOTO 2660
2610 IF LEN(F$)<2 THEN PRINT E4$: GOTO 2580
2620 E=VAL(LEFT$(F$,1)): IF E<1 OR E>5 THEN PRINT E5$: GOTO2580
2630 F=1: F$=MID$(F$,2)
2640 IF F$<>N$(F) THEN F=F+1: IF F<13 THEN 2640
2650 IF F>12 THEN PRINT E4$: GOTO 2580
2660 IF EM=0 THEN T(0)=T(0)+1 ELSE B(0)=B(0)+1
2670 FOR X=L TO N STEP -1
2680 IF EM=0 THEN T(X+1)=T(X) ELSE B(X+1)=B(X)
2690 NEXT X
2700 P=4096*E + 256*F + D
2710 IF EM=0 THEN T(N)=P ELSE B(N)=P
2720 BF=0: L=L+1: N=N+1: GOTO 2580
2730 '
2740 '**** DELETE NOTES
2750 CLS
2760 P=0: Q=0: INPUT"DELETE FROM,TO";P,Q
2770 IF P<1 OR P>Q OR Q>L THEN 2760
2780 IF EM=0 THEN T(0)=T(0)-(Q-P+1) ELSE B(0)=B(0)-(Q-P+1)
2790 BF=0
2800 IF Q=L THEN RETURN
2810 FOR X=Q+1 TO L
2820 IF EM=0 THEN T(P)=T(X) ELSE B(P)=B(X)
2830 P=P+1: NEXT X: RETURN
2840 '
2850 '**** DISPLAY NOTES
2860 CLS
2870 IF EM=0 THEN PRINT"TREBLE" ELSE PRINT"BASE"
2880 PRINT
2890 PRINT"   DURATION,NOTE"
2900 FOR X=1 TO L: PRINT USING "####:";X;
2910 IF EM=0 THEN D=T(X) AND 255 ELSE D=B(X) AND 255
2920 IF EM=0 THEN E=T(X)/4096 ELSE E=B(X)/4096
2930 IF EM=0 THEN F=(T(X) AND 3840)/256
              ELSE F=(B(X) AND 3840)/256
2940 F$=N$(F)
2950 IF E>0 THEN PRINT USING "#####  #%% ";D;E;F$
             ELSE PRINT USING "#####   %% ";D;F$
2960 NEXT X
2970 PRINT: INPUT"PRESS ENTER";M: RETURN
2980 '
```

Listing 1 continues

Listing 1 continued

```
2990 '**** PRINT NOTES
3000 CLS
3010 M=0: INPUT"PRINT HOW MANY COLUMNS";M
3020 IF M<1 THEN 3010
3030 LPRINT:LPRINT
3040 IF EM=0 THEN LPRINT"TREBLE" ELSE LPRINT"BASE"
3050 LPRINT
3060 FOR X=1 TO M: LPRINT"   DURATION,NOTE";: NEXT
3070 LPRINT
3080 P!=L/M: Q=L/M: IF P!>Q THEN Q=Q+1
3090 FOR X=1 TO Q
3100 FOR N=0 TO M-1: P=X+N*Q: IF P>L THEN 3160
3110 IF EM=0 THEN D=T(P) AND 255 ELSE D=B(P) AND 255
3120 IF EM=0 THEN E=T(P)/4096 ELSE E=B(P)/4096
3130 IF EM=0 THEN F=(T(P) AND 3840)/256
          ELSE F=(B(P) AND 3840)/256
3140 F$=N$(F)
3150 IF E>0 THEN LPRINT USING "####:#####   #%% ";P;D;E;F$;
         ELSE LPRINT USING "####:#####    %% ";P;D;F$;
3160 NEXT N: LPRINT: NEXT X: RETURN
3170 '
3180 '**** BUILD PLAY BUFFER
3190 CLS
3200 PRINT"BUILDING PLAY BUFFER":PRINT
3210 PRINT"","TREBLE","BASE"
3220 PRINT "LENGTH",T(0),B(0)
3230 T=1: B=1: X=1: GOSUB 3440: GOSUB 3480
3240 D(X+1)=C2*2[(5-TE) + N(TF)
3250 D(X+2)=C2*2[(5-BE) + N(BF)
3260 D(X+3)=(P AND 2) + C2*(Q AND 1)
3270 PRINT@256,,T,B
3280 IF TD>=BD THEN 3320
3290 D(X)=TD: T=T+1: BD=BD-TD
3300 IF T<=T(0) THEN GOSUB 3440
3310 GOTO 3390
3320 IF TD=BD THEN 3360
3330 D(X)=BD: B=B+1: TD=TD-BD
3340 IF B<=B(0) THEN GOSUB 3480
3350 GOTO 3390
3360 D(X)=BD: B=B+1: T=T+1
```

Listing 1 continues

bass lists are not affected.

Understanding the Program

The most important variables in the Basic program, together with their meanings, are listed in Table 2.

The parameters for each treble or bass note are packed into one integer by the formula: 4096 * octave + 256 * note + duration. Figure 2 shows this format.

The play buffer consists of a note count followed by 8-byte entries. The format for each entry is shown in Table 3. Each note change has a new entry.

The USR subroutine reads the play buffer to produce the tones. Listing 2 is an Assembly-language form of the USR subroutine. Line 130 gets the play buffer address. Lines 160–170 transfer this address into the IX register. The A' register maintains the current tone command value. Lines 330–370 and 580–620 count the treble, bass, and note duration values until one of them becomes due.

When the treble count becomes due, lines 890–980 reload the count and continue timing. When the treble octave count becomes due, lines 990–1050 change the tone level command value and resend the command. Lines

400–560 update the bass note counts in the same way.

When the duration count becomes due, lines 750–860 advance IX to point to the next note entry in the play buffer, terminate the treble and bass octave counts, and continue timing, allowing the treble and bass timers to complete the current cycle. Control returns to Basic when all notes have been played.

Enhancing the Program

There are many ways to improve the program. Adding more editing features would make it easier to enter and modify the music—or including a subroutine facility would make it possible to play longer pieces. The greatest improvement, however, would be a hardware change.

The TRS-80 clock is not fast enough to produce the high quality of sound that best enhances the dual-voice technique. Speeding it up would make the timing of each note more accurate and would enable the program to generate higher-quality notes and increase the range of tones.

This program is just one method for dual-voice synthesis. Another example is synthesizing voices out of the cassette port. With new innovations and experi-

Listing 1 continued

```
3370 IF T<=T(Ø) THEN GOSUB 3440
3380 IF B<=B(Ø) THEN GOSUB 3480
3390 D(X)=D(X)-C2: X=X+4
3400 IF T<=T(Ø) AND B<=B(Ø) THEN 3240
3410 D(Ø)=X/4: BF=-1: RETURN
3420 '
3430 '**** COMPUTE TREBLE PLAY BUFFER ENTRY
3440 TD=C4/(T(T) AND C1): TE=T(T)/C5
3450 TF=(T(T) AND C3)/C2: P=TF>Ø: RETURN
3460 '
3470 '**** COMPUTE BASE PLAY BUFFER ENTRY
3480 BD=C4/(B(B) AND C1): BE=B(B)/C5
3490 BF=(B(B) AND C3)/C2: Q=BF>Ø: RETURN
3500 '
3510 '**** CHANGE TIMING
3520 CLS
3530 M!=1: INPUT"TIMING MULTIPLIER";M!
3540 FOR X=1 TO D(Ø)*4 STEP 4
3550 D(X)=(D(X) AND 255)*M! - 256
3560 NEXT
3570 RETURN
3580 '
3590 '**** PLAY MUSIC
3600 CLS
3610 PRINT"<1> PLAY USING DISK BASIC"
3620 PRINT"<2> PLAY USING LEVEL II BASIC"
3630 PRINT
3640 M=Ø: INPUT"SELECTION";M
3650 IF M<1 OR M>2 THEN 3640
3660 CLS: Q=VARPTR(D(Ø))
3670 IF M=2 GOTO 3720
3680 P!=PEEK(VARPTR(M$)+1)+256*PEEK(VARPTR(M$)+2)
3690 IF P!>32767 THEN P!=P!-65536
3700 P=P!: DEFUSRØ=P: M=USRØ(Q)
3710 RETURN
3720 POKE 16526,PEEK(VARPTR(M$)+1)
3730 POKE 16527,PPEK(VARPTR(M$)+2)
3740 M=USR(Q)
3750 RETURN
3760 '
3770 'END OF PROGRAM
```

mentation, perhaps someone will even discover a way to produce the sounds of an entire orchestra. ■

Lee Morgenstern can be reached at 14358 Dyer St., Sylmar, CA 91342.

Program Listing 2

```
                        ;DUAL-VOICE MUSIC USR SUBROUTINE
                        ; BY LEE MORGENSTERN, MAY, 1982
                        ;
0000 CD7F0A                     CALL    0A7FH           ;HL=NOTE LIST ADDRESS
0003 F3                         DI                      ;DISABLE INTERRUPTS
0004 EB                         EX      DE,HL           ;DE=NOTE LIST ADDRESS
0005 DD210000                   LD      IX,0            ;
0009 DD19                       ADD     IX,DE           ;IX=NOTE LIST ADDRESS
000B DD4E00                     LD      C,(IX)          ;BC=NOTE LIST LENGTH
000E DD4601                     LD      B,(IX+1)        ;
0011 DD23                       INC     IX              ;
0013 DD23                       INC     IX              ;
0015 110800                     LD      DE,8            ;NOTE LIST ENTRY LENGTH
0018 D9                         EXX                     ;
0019 3E04                       LD      A,4             ;SET CASSETTE COMMAND
001B 08                         EX      AF,AF'          ;SAVE TONE LEVEL
001C DD6600                     LD      H,(IX)          ;GET NOTE DURATION
001F DD6E01                     LD      L,(IX+1)        ;
0022 DD4E02                     LD      C,(IX+2)        ;SET TREBLE COUNTER
0025 DD4603                     LD      B,(IX+3)        ;SET TREBLE OCTAVE
0028 DD5E04                     LD      E,(IX+4)        ;SET BASE COUNTER
002B DD5605                     LD      D,(IX+5)        ;SET BASE OCTAVE
002E 3E06               MUSIC1  LD      A,6             ;SET TIMING COUNTER
0030 3C                 MUSIC2  INC     A               ;UPDATE TIMING COUNTER
0031 B9                         CP      C               ;CHECK TREBLE DUE TIME
0032 3048                       JR      NC,MUSIC6       ;JUMP IF DUE
0034 BB                 MUSIC3  CP      E               ;CHECK BASE DUE TIME
0035 381C                       JR      C,MUSIC4        ;JUMP IF NOT DUE
                                ;
                                ;RELOAD BASE TIMER
0037 79                         LD      A,C             ;GET TREBLE TIMER
0038 93                         SUB     E               ;COMPUTE NEXT DUE TIME
0039 4F                         LD      C,A             ;
003A 7D                         LD      A,L             ;GET DURATION TIMER
003B 93                         SUB     E               ;COMPUTE NEXT DUE TIME
003C 6F                         LD      L,A             ;
003D DD5E04                     LD      E,(IX+4)        ;RELOAD BASE TIMER
0040 3E02                       LD      A,2             ;SET TIMING COUNTER
0042 15                         DEC     D               ;DECREMENT OCTAVE COUNTER
0043 200E                       JR      NZ,MUSIC4       ;JUMP IF NOT DONE
0045 08                         EX      AF,AF'          ;GET PREVIOUS TONE LEVEL
0046 DDAE07                     XOR     (IX+7)          ;TOGGLE BASE TONE LEVEL
0049 D3FF                       OUT     (255),A         ;COMMAND TONE LEVEL
004B 08                         EX      AF,AF'          ;SAVE TONE LEVEL
004C DD5605                     LD      D,(IX+5)        ;RELOAD OCTAVE COUNTER
004F 3E05                       LD      A,5             ;SET TIMING COUNTER
0051 1804                       JR      MUSIC5          ;CONTINUE
                                ;
0053 3C                 MUSIC4  INC     A               ;UPDATE TIMING COUNTER
0054 B9                         CP      C               ;CHECK TREBLE DUE TIME
0055 3025                       JR      NC,MUSIC6       ;JUMP IF TREBLE DUE
0057 BD                 MUSIC5  CP      L               ;CHECK NOTE DURATION
0058 38D6                       JR      C,MUSIC2        ;JUMP IF NOT DUE
                                ;
                                ;RELOAD NOTE DURATION TIMER
005A 79                         LD      A,C             ;GET TREBLE TIMER
005B 95                         SUB     L               ;COMPUTE NEXT DUE TIME
005C 4F                         LD      C,A             ;
005D 7B                         LD      A,E             ;GET BASE TIMER
005E 95                         SUB     L               ;COMPUTE NEXT DUE TIME
005F 5F                         LD      E,A             ;
0060 DD6E01                     LD      L,(IX+1)        ;RELOAD DURATION TIMER
0063 3E02                       LD      A,2             ;SET TIMING COUNTER
0065 25                         DEC     H               ;DECREMENT DURATION COUNT
0066 20C8                       JR      NZ,MUSIC2       ;JUMP IF NOT DONE
0068 D9                         EXX                     ;
0069 DD19                       ADD     IX,DE           ;NEXT NOTE
006B 0B                         DEC     BC              ;
006C 78                         LD      A,B             ;
006D B1                         OR      C               ;
006E C8                         RET     Z               ;RETURN IF NO MORE NOTES
006F D9                         EXX                     ;
0070 0601                       LD      B,1             ;TERMINATE TREBLE
0072 1601                       LD      D,1             ;TERMINATE BASE
0074 DD6600                     LD      H,(IX)          ;GET NEXT NOTE DURATION
0077 DD6E01                     LD      L,(IX+1)        ;
007A 18B2                       JR      MUSIC1          ;RESTART PLAY TIMING
                                ;
                                ;RELOAD TREBLE TIMER
007C 7B                 MUSIC6  LD      A,E             ;GET BASE TIMER
007D 91                         SUB     C               ;COMPUTE NEXT DUE TIME
007E 5F                         LD      E,A             ;
007F 7D                         LD      A,L             ;GET DURATION TIMER
0080 91                         SUB     C               ;COMPUTE NEXT DUE TIME
0081 6F                         LD      L,A             ;
0082 DD4E02                     LD      C,(IX+2)        ;RELOAD TREBLE TIMER
0085 3E03                       LD      A,3             ;SET TIMING COUNTER
0087 05                         DEC     B               ;DECREMENT OCTAVE COUNTER
0088 20AA                       JR      NZ,MUSIC3       ;JUMP IF NOT DONE
008A 08                         EX      AF,AF'          ;GET PREVIOUS TONE LEVEL
008B DDAE06                     XOR     (IX+6)          ;TOGGLE TREBLE TONE LEVEL
008E D3FF                       OUT     (255),A         ;COMMAND TONE LEVEL
0090 08                         EX      AF,AF'          ;SAVE TONE LEVEL
0091 DD4603                     LD      B,(IX+3)        ;RELOAD OCTAVE COUNTER
0094 3E05                       LD      A,5             ;SET TIMING COUNTER
0096 189C                       JR      MUSIC3          ;CONTINUE
                                ;
0000                            END
00000 TOTAL ERRORS
```

LOAD 80

FlexCat, Your Basic DBM

by Lawrence A. Terre

Wouldn't it be nice to use one data-base manager for all your filing and retrieval needs? FlexCat is just what fills the bill.

Tired of rewriting your data-base program for each application? Forget keywords for a file created last year? FlexCat is a flexible text-oriented cataloging system able to search data using partial or limited information.

Before FlexCat, locating a specific record in a large file was tedious, especially if my recollection of that record was fragmented. FlexCat's search feature provides a method to extract a record or records based on partial information.

Initialization Program

This program creates a disk file according to parameters determined by your responses to prompts. The file created will contain heading, field, and data information to be used by the two remaining programs. A test record is also created.

> *The Key Box*
> **Disk Basic**
> **Model I**
> **48K RAM**
> **TRSDOS**
> **One Disk Drive**
> **Line Printer I**

The fixed-format screen layout displays one record at a time. Headings appear left-justified in columns 1–21, exactly as input, with trailing periods as fillers. When entered from the main program, the data appears in fields (columns 22–62). Each heading must have at least one field and the total number of fields cannot exceed 14.

The program next asks if the heading's information will be numeric or alphanumeric. Incorrectly responding to this question affects the way the data is evaluated in the sort program, so be careful.

You will then be given the option to redefine parameters or create a disk file. When creating, you can supply your own file name or use the default (INITDISK). For easy file identification in the directory, FlexCat automatically adds the extension /TXT to all file names. To prevent accidental destruction of a preexisting file the program checks and informs you if a file with the same name already exists. You are given the option to create a new file under the old name or select a different name.

Main Program

The main program allows you to review, manipulate, print, and otherwise interact directly with the file data. When this program runs, the load function executes first using the same default file name as the disk-initialization program (unless you assign a new default name).

The main program's four major routines are displayed on the menu: Add, List, Search, and Save. The legal options, available within a given routine, are displayed on the lines immediately preceding and following the displayed record.

Legal Options

The Add module is used to input additional file records. After selecting this function, the heading and field parameters input during disk initialization appear on the screen to cue you. The allotted field space is indicated by a series of graphic blocks. When the cursor appears at the beginning of the first field, enter that data. The cursor will not advance beyond the first field unless some data (even a space) is input.

Then it is possible to move to the next or any other field by pressing enter. To terminate this Add loop, type "end" at the beginning of any field. The current record will not be added to the file.

You can sequentially access records within a file by using the List routine. There are other options available using List. Pressing the up arrow permits you to access the previous record. Pressing shift, D graphically marks that record to indicate it will not be transferred to the new file during the save. You can salvage records marked for deletion by retyping the first field when in the update mode.

Also, deleted records can be located via the search. In a similar fashion, the

shift, U allows you to update the displayed record (only fields requiring modification need to be retyped) and shift, P prints the current record. The latter option is ignored if the printer is not ready. Pressing shift, S skips past the next nine records to display the tenth, and shift, L displays the last record of the file. Pressing E brings you back to the menu.

The Search option locates specific records within the file via a text search of any fields you choose. The text search scans all fields attached to a specific heading for the occurrence of a character, word, or phrase anywhere within the field. This routine permits you to carry out up to five concurrent text searches within individual or between multiple fields. In other words, you can implement any combination of tests ranging from five separate tests on one field to individual tests on each of five fields.

"You always have the option of exiting to the menu by pressing the E key."

A record displays only when it satisfies the requirements of all specified tests. Of course, you always have the option of performing fewer than five tests. If no search heading is entered for the first prompt, a default search of the records marked for deletion commences. Legal options available in this routine include update, print, delete, and escape.

Save transfers all records not marked for deletion to a file with either the default name (file name used to load data) or a new name.

The menu lists two additional options: enable auto print and end. The first allows the automatic printing of either the entire file or only records satisfying the search criteria. A command to print in the absence of a ready printer will not cause the system to hang up indefinitely. Within the auto-print operation, you always have the option of exiting to the menu by pressing the E key. The end function displays the number of records in memory not marked for deletion and provides a final opportunity to return to the menu (possibly to save your data).

Sort Program

FlexCat's sort is an adaptation of D. Walker's "Beyond Shell Metzner" (*80 Micro*, September 1980). Although the FlexCat sort program does not influence the operation of either the disk-initialization program or main program, it enhances the system by providing a method of organizing the file according to parameters you supply.

When running this program, it asks you to provide a file name. The program adds /TXT to the file name and informs you if the file is not on the disk.

Program Listing 1. Initialization

```
10 '-------------------------------------------------------------
15 '
20 'FLEXCAT INITIALIZATION PROGRAM
25 '
30 'WRITTEN BY:
35 '           LAWRENCE A. TERRE
40 '           1100 RANDVILLE DRIVE #205
45 '           PALATINE, ILLINOIS 60067
55 '-------------------------------------------------------------
100 CLEAR2000: DIMD$(500): DIMB$(14,3): F$="INITDISK"
110 D$="TEST RECORD FROM INITIALIZATION PROGRAM"
120 B$=STRING$(21,".")
130 CLS: A$="DISK INITIALIZATION PROGRAM": GOSUB570
140 TF=14: H=0: FORX=1TO500: NEXT
145 '---INPUT PARAMETERS
150 FORX=1TO14
160     CLS:
        PRINT
170     A$="HEADINGS MUST NOT EXCEED 21 CHARACTERS.":
        GOSUB570  :
        PRINT
180     A$="AFTER HEADING # 1, ANY BLANK HEADING WILL":
        GOSUB570
190     A$="TERMINATE THE ENTRY PROCESS":
        GOSUB570
195 '---ENTER HEADING NAME
200     PRINT"HEADING #";X;" WILL BE ";:PRINT,":";
210     LINEINPUTB$(X,1):
        L=LEN(B$(X,1))
220     IFL>21 OR (L=0ANDX=1)
            GOTO200
230     IFL=0 AND X<>1
            GOTO340
235     H=H+1
240     B$(X,1)=B$(X,1)+RIGHT$(B$,21-L)
245 '---ENTER DATA TYPE (NUMERIC/ALPHANUMERIC)
250     PRINT"NUMERIC OR ALPHA - ( N OR A )";:PRINT,":";
260     LINEINPUTB$(X,2):
        B$(X,2)=LEFT$(B$(X,2),1)
270     IFB$(X,2)<>"N" AND B$(X,2)<>"A"
            THEN250
280     PRINT"NUMBER OF FIELD OUT OF";TF;:PRINT,":";
285 '---ENTER NUMBER OF FIELDS FOR THIS HEADING
290     LINEINPUTB$(X,3)
300     IFVAL(B$(X,3))<1 OR VAL(B$(X,3))>TF
            GOTO280
310     TF=TF-VAL(B$(X,3))
320     IFTF=0GOTO340
330 NEXT
340 CLS: A$="PRESS <D>ONE OR <R>EDEFINE": GOSUB570  : GOSUB580

350 IFZZ$="R"GOTO130
360 IFZZ$="D"GOTO380
370 GOTO340
375 '---CREATE DISK FILE
380 CLS: A$="CREATE FILE ROUTINE": GOSUB570
390 PRINT: PRINT"DEFAULT FILE NAME IS <";F$;">": PRINT
400 INPUT"IF DESIRED, ENTER NEW FILE NAME W/O EXTENSION ";F$
410 ON ERROR GOTO 490
415 '---TEST FOR EXISTING FILE WITH SAME NAME
420 OPEN"I",1,F$+"/TXT": CLOSE
430 PRINT
440 A$="FILE ALREADY EXISTS.  PRESS <C> TO CREATE": GOSUB570
450 A$="OR <R> TO RENAME FILE": GOSUB570  : GOSUB580
460 IFZZ$="R"GOTO390
470 IFZZ$="C"GOTO490
480 GOTO460
485 '---FILE NAME SELECTED IS NOT ON DISKETTE
```

Listing 1 continues

After the entry of a valid file name, all headings read from the file are displayed on the screen. You are asked to rank in order from major to minor the headings and specify sort directions (ascending/descending).

The program automatically assigns the data type (numeric/alphanumeric) to all fields based on information supplied during the disk-initialization program. The data type you selected during initialization will now affect the collating sequence. You are asked for intermediate through minor headings until all have been entered or you input "stop."

Finally, a prompt asks you to insert the destination disk and press enter. As the file sorts in the specified order, the screen displays the field number being sorted as well as a graphic character for every swap made. The time required for the sort and the number of records sorted are displayed and the file is written on the disk under the same name that was used to read it.

The Program Listings

All programs are well documented with comments. When typing in the main and sort programs you can omit the comments and unnecessary spaces. Be sure to adjust the number in the Clear statement of the main program to ensure that all unallocated memory space gained is reserved for string storage.

The sizes of the main and sort programs without comments and unnecessary spaces are approximately 5K and 4K respectively. To facilitate program entry for those who use the Basic automatic line-numbering feature, I started all the necessary program lines at 100, incrementing by 10.

Compressing the code also results in the added benefit of faster execution time, which becomes especially important in search and sort operations.

Applications

I have used this program for a wide variety of applications ranging from keeping track of magazine articles to cataloging taped class lectures, videocassettes, items for a home inventory, and recipes. The ability to perform multiple text searches on each record while retrieving only the one satisfying all criteria has been an especially helpful aid. You should have no difficulty finding uses for FlexCat. ■

Lawrence Terre lives at 1101 West Thatch # 16, Auburn University, AL 36849.

Listing 1 continued

```
490 RESUME500
500 ON ERROR GOTO 0     : F=14-TF: NR=1
510 OPEN"O",1,F$+"/TXT"
520 PRINT#1,NR;F;H
525 '---WRITE HEADINGS TO DISK
530 FORCC=1TOH:
          PRINT#1,
          CHR$(34);B$(CC,1);CHR$(34);
          CHR$(34);B$(CC,2);CHR$(34);
          CHR$(34);B$(CC,3);CHR$(34):
    NEXT
535 '---WRITE TEST DATA TO DISK
540 FORCC=1TOF
550       PRINT#1,CHR$(34);D$;CHR$(34)
560 NEXT: CLOSE: END
565 '---CENTER AND PRINT A$ ROUTINE
570 TB=.5 * LEN(A$): TB=32-TB: PRINTTAB(TB)A$: RETURN
575 '---INKEY$ ROUTINE
580 ZZ$=INKEY$:
    IFZZ$=""
          THEN580
    ELSE
          RETURN
590 END
```

Program Listing 2. Sort

```
10  '----------------------------------------------------------------
15  'FLEXCAT SORT PROGRAM
20  '
25  'WRITTEN BY:
30  '                     LAWRENCE A. TERRE
35  '                     1100 RANDVILLE DRIVE #205
40  '                     PALATINE, ILLINOIS 60067
45  '
50  '----------------------------------------------------------------
55  '---TO CONSERVE MEMORY SPACE, REMOVE ALL UNNECESSARY--------
60  '---SPACES AND DELETE ALL REMARKS AS THERE ARE NO----------
65  '---BRANCHES TO THESE STATEMENTS.---------------------------
80  '----------------------------------------------------------------
100 CLEAR 26000
110 DEFINT B-Z
120 CLS: PRINTTAB(12)"INPUT DATA FROM DISK ROUTINE"
130 A1=0: A2=0: A3=0: A4=0
140 ON ERROR GOTO 1090
150 INPUT"ENTER FILE NAME ";F$: OPEN"I",1,F$+"/TXT"
160 ON ERROR GOTO 0
170 INPUT#1, NR, F, H
180 DIM C$(NR,F): DIM A$(F+1): DIM B$(F,3):
    DIM C1(F+1): DIM C2(F+1)
190 DIM D(F+1): Q=1: Y=1
200 FOR X=1 TO F+1:
        A$(X)="": C1(X)=-1: C2(X)=0: D(X)=0:
    NEXTX
205 '---PUT IMAGE OF HEADINGS IN ARRAY A$( )
210 FOR X=1 TO H
220     INPUT#1, B$(X,1), B$(X,2), B$(X,3)
230     A$(Y)= B$(X,1):
        Y= Y+VAL(B$(X,3))
240 NEXTX
245 '---PUT DATA IN ARRAY C$( , ) AND REMOVE SINGLE SPACES
250 FOR X=1 TO NR:
        FOR Y=1 TO F
260         INPUT#1, C$(X,Y)
270         IF C$(X,Y)= " " THEN C$(X,Y)= ""
280     NEXTY:
    NEXTX:
    CLOSE
285 '---DISPLAY HEADINGS ON CRT
290 CLS:
    FOR X=1 TO H STEP2:
        PRINT B$(X,1);:
        PRINTTAB(40) B$(X+1,1):
    NEXTX
300 PRINT STRING$(63,"-")
305 '---INPUT SORT FIELD HEADING
310 FOR XX=1 TO H
320     PRINT"ENTER HEADING OR <STOP>.  PRIORITY #;" XX; "? ";
330     LINEINPUT S$
335 '---TEST FOR VALID FIELD HEADING
```

Listing 2 continues

Listing 2 continued

```
      340     IF S$= "STOP" AND XX>1
                THEN450
      350     L= LEN(S$):
                S$= S$  + STRING$(21-L,".")
      360     FOR X=1 TO H:
                IF S$= B$(X,1)
                GOSUB460  :
                GOTO390
370     NEXTX
380     PRINT"NON-EXISTANT HEADING":
        GOTO320
385 '---INPUT SORT DIRECTION (ASCENDING OR DECENDING)
390     PRINT"SORT ORDER <A>SCENDING OR <D>ECENDING   ? ";
400     LINEINPUT D$:
        D$= LEFT$(D$,1)
410     IF D$ <> "A" AND D$ <>"D"
          THEN390
420     IF D$= "A"
          THEN AD= 1
        ELSE
          AD= 2
430     GOSUB490
440 NEXTXX
450 CLS:
    INPUT"INSERT DESTINATION DISKETTE, PRESS ENTER"; U$:
    GOTO 580
455 '---CHECK DATA TYPE.  IF B$( ,2)= "N" THE DATA IS
456 '---NUMERIC ELSE DATA TYPE IS ALPHANUMERIC
460 IF B$(X,2)= "N"
        THEN AN= 1
    ELSE
        AN= 2
465 '---CHECK NUMBER OF FIELDS ASSIGNED TO HEADING
470 FL= VAL(B$(X,3))
480 RETURN
485 '---LOOK FOR SORT HEADING IN ARRAY A$( )
490 FOR V= 1 TO F
500     IF S$= A$(V) THEN 520
510 NEXT V:
    RETURN
515 '---FOUND IT.  V= STARTING FIELD POSITION OF HEADING
516 '---BUILD PARAMETER TABLE.  C1( )= FIELD SORT ORDER
517 '---WHEN THE FIRST (-1) IS FOUND IN C1( ), THE SORT
518 '---IS DONE.  C2( )= 1 FOR NUMERIC AND 2 FOR A/N DATA
519 '---D( )= 1 FOR ASCENDING AND 2 FOR DECENDING SEQUENCE
520 FOR N= Q TO Q+FL-1
530     C1(N)= V:
        C2(N)= AN:
        D(N)= AD
540     V= V+1
550 NEXT N
560 Q= Q+FL
570 RETURN
580 Z6= 1: J= NR: A= NR: K= 1 :
    IF NR<2 THEN END
590 POKE &H4041, 00: POKE &H4042, 00: POKE &H4043, 00
594 '-------------------------------------------------------
595 '---THE FOLLOWING HAS BEEN ADAPTED FROM-------------------
596 '---D. WALKER, "BEYOND SHELL METZNER", 80 MICRO, 9/80------
597 '-------------------------------------------------------
600 CLS
605 '---SORTS USING SHELL METZNER ALGORITHM
610 Z3= J
620 Z3= INT(Z3/2):
    IF Z3= 0
        THEN 720
630 CLS: PRINT@20,"SORTING FIELD #"; C1(Z6)
640 Z4= K: Z5= J-Z3
650 Z7= Z4
660 Z8= Z7+Z3
670 IF Z6 > 1 AND (Z7 < K OR Z7 > J OR Z8 < K OR Z8 > J )
        GOTO 710
680 GOSUB950
690 IF F1= 2
        GOTO 710
    ELSE
        GOSUB 1040
700 Z7= Z7-Z3:
    IF Z7 < 1
        THEN 710
    ELSE
        660
710 Z4= Z4+1:
    IF Z4 > Z5
        THEN 620
    ELSE
        650
720 IF Z6= 1 THEN 740
730 IF Z6 > 1 AND M= A
        THEN 740
    ELSE
        760
735 '---SEE IF THERE IS ANOTHER FIELD TO SORT
740 Z6= Z6+1:
    IF C1(Z6)= -1
        THEN 890
745 '---SORT SEGMENT COMPUTATION ON INTERMEDIATE THRU
746 '---MINOR FIELDS
750 M= 1
760 Z7= M: Z8= M+1: J= 1: L= 0
770 FOR X1= Z6-1 TO 1 STEP -1
780     IF C2(X1)= 2:
            IF C$( Z7, C1(X1)) < > C$( Z8, C1(X1)):
                L= 1:
                GOTO 820
790     IF C2(X1)= 1:
            IF VAL( C$( Z7, C1(X1))) < > VAL( C$( Z8, C1(X1))):
                L= 1:
                GOTO 820
800 NEXT X1
810 J= J+1
820 IF L= 0 THEN 850
830 IF L= 1 AND J > 1
        THEN K= M:
        J= Z7:
        M= Z8:
        GOTO 610
840 L= 0: M= Z8
850 Z7= Z7+1: Z8= Z8+1
860 IF Z7 < A THEN 770
870 IF L= 0
        THEN K= M:
        M= Z7:
```

Listing 2 continues

Listing 2 continued

```
        J= Z7:
        GOTO 610
880 GOTO 740
890 ZZ$= RIGHT$( TIME$, 9):
    PRINT: PRINT"SORT TIME FOR"; NR; "RECORDS - "; ZZ$
900 OPEN"O",1,F$+"/TXT"
910 PRINT#1, NR; F; H
920 FOR CC= 1 TO H:
    PRINT#1, CHR$(34); B$(CC,1); CHR$(34);
             CHR$(34); B$(CC,2); CHR$(34);
             CHR$(34); B$(CC,3); CHR$(34):
    NEXT
930 FOR X= 1 TO NR:
        FOR Y= 1 TO F:
            PRINT#1, CHR$(34); C$(X,Y); CHR$(34):
        NEXTY:
    NEXTX
940 CLOSE: END
945 '---RECORD ADDRESS POINTER SWITCH CHECK
950 IF D(Z6)= 2 GOTO 990
960 IF C2(Z6)= 1GOTO 980
970 IF C$(Z7,C1(Z6)) > C$(Z8,C1(Z6)):
        GOTO 1020
    ELSE
        GOTO 1030
980 IF VAL(C$(Z7,C1(Z6))) > VAL(C$(Z8,C1(Z6)))
        GOTO 1020
    ELSE
        GOTO 1030
990 IF C2(Z6)= 1 THEN 1010
1000 IF C$(Z7,C1(Z6)) < C$(Z8,C1(Z6))
        GOTO 1020
    ELSE
        GOTO 1030
1010 IF VAL(C$(Z7,C1(Z6))) < VAL(C$(Z8,C1(Z6)))
         GOTO 1020
     ELSE
         GOTO 1030
1020 F1= 1: RETURN
1030 F1= 2: RETURN
1035 '---RECORD ADDRESS POINTER SWITCH
1040 PRINTCHR$(153);:
     FOR C= 1 TO F:
         FOR SZ= 0 TO 2
1050             A3= VARPTR(C$(Z7,C))+SZ:
                 IF A3 > 32767:
                     A1= (PEEK(A3) - 65536):
                     A3= A3 - 65536:
                 ELSE
                     A1= PEEK(A3)
1060             A4= VARPTR(C$(Z8,C)) + SZ:
                 IF A4 > 32767:
                     A2= (PEEK(A4) - 65536):
                     A4= A4 - 65536:
                 ELSE
                     A2= PEEK(A4)
1070             POKE A3, A2: POKE A4, A1
1080         NEXT SZ:
     NEXT C:
     RETURN
1090 PRINT:PRINT,"FILE NOT ON DISKETTE":PRINT:RESUME150
1100 END
```

Program Listing 3. Main Program

```
10 '-----------------------------------------------------------
15 ' FLEXCAT MAIN PROGRAM
20 '
25 'WRITTEN BY:
30 '           LAWRENCE A. TERRE
35 '           1100 RANDVILLE DRIVE #205
40 '           PALATINE, ILLINOIS 60067
45 '
50 '-----------------------------------------------------------
55 'REMARK STATEMENTS AND UNNECESSARY SPACES SHOULD BE
60 'REMOVED TO CONSERVE MEMORY.  THERE ARE NO PROGRAM
65 'BRANCHES TO REMARK STATEMENTS.  IF REMARKS AND SPACES
70 'ARE REMOVED, THE NUMBER IN THE CLEAR STATEMENT SHOULD
75 'BE INCREASED.
80 'NOTE:  [ DENOTES UP ARROW
85 '-----------------------------------------------------------
100 POKE16425,1: CLEAR20000: DEFINTA-Z: DIMD$(2100): DIMB$(14,3)

110 F$="INITDISK": NR=0: II=0: B$=STRING$(21,"."):
    Z$=STRING$(41,CHR$(136))+CHR$(32)
120 ON ERROR GOTO 210
125 '---READ DISK ROUTINE--------------------------------------
130 CLS: A$="MAIN PROGRAM": GOSUB1310
140 PRINT: PRINT"DEFAULT FILE NAME IS <";F$;">"
150 INPUT"IF DESIRED, ENTER NEW FILE NAME ";F$
155 '---TEST FOR EXISTING FILE WITH SAME NAME
160 OPEN"I",1,F$+"/TXT"
165 '---NO EXISTING FILE, TURN OFF ERROR HANDLING ROUTINE
170 ON ERROR GOTO 0
175 '---NR= # OF RECORDS, F= # OF FIELDS, h= # OF HEADINGS
180 INPUT#1,NR,F,H: II=NR
185 '---B$( ,1)=HEADING, B$( ,2)="N"UMERIC / ="A"LPHANUMERIC
186 '---B$( ,3)= # OF FIELDS FOR HEADING B$( ,1)
190 FORCC=1TOH:
        INPUT#1,B$(CC,1),B$(CC,2),B$(CC,3):
    NEXT
195 '---ZZ=SUBSCRIPT FOR LAST FIELD OF LAST RECORD
196 '---D$( )=DATA STRING
200 ZZ=(NR*F):
    FORCC=1TOZZ:
        INPUT#1,D$(CC):
    NEXT:
    CLOSE: GOTO220
210 PRINT: A$="FILE DOES NOT EXIST": GOSUB1310 : RESUME140
215 '---MENU SELECTION ROUTINE---------------------------------
220 CLS: PRINT: BT=20:
    A$="FLEXCAT: FLEXIBLE TEXT-ORIENTED CATALOGING SYSTEM":
    GOSUB1310
230 PRINT: PRINTTAB(BT)"1 - ADD DATA"
240 PRINTTAB(BT)"2 - LIST DATA"
250 PRINTTAB(BT)"3 - SEARCH DATA"
260 PRINTTAB(BT)"4 - SAVE DATA ON DISKETTE"
270 PRINTTAB(BT)"5 - ENABLE AUTO PRINT FUNCTION"
280 PRINTTAB(BT)"6 - END PROGRAM"
290 PRINT: A$="MAKE SELECTION, THEN PRESS <ENTER>": GOSUB1310
300 ZZ$="2": INPUTZZ$: ZZ=VAL(ZZ$): IFZZ<0 OR ZZ>6THEN220
310 ON ZZ GOSUB330  ,500  ,750  ,630  ,1450 ,1120
320 GOTO220
```

Listing 3 continues

Listing 3 continued

```
325 '---ADD DATA ROUTINE--------------------------------------------
330 CLS:
    A$="*INPUT*   TYPE 'END' TO KILL PRESENT RECORD AND EXIT":
    GOSUB1310
335 '---PRINT HEADINGS & GRAPHIC BOXES FOR FIELDS
340 FORX=1TOH:
        PRINTB$(X,1);
350     FORY=1TOVAL(B$(X,3)):
            PRINTTAB(22)Z$;:
        NEXTY:
    NEXTX
360 PRINTSTRING$(63,"-");:T=22:Z=(II*F)+1
365 '---REPRINT HEADING & INPUT FIELD DATA
370 FORX=1TOH:
        PRINT@T+42,B$(X,1);
380     FORY=1TOVAL(B$(X,3)):
            T=T+64
390         PRINT@T,Z$;:
            PRINT@T,"";
400         LINEINPUTBB$:
            BB$=LEFT$(BB$,41)
410         IFBB$="END"THEN490
420         IFINT((Z-1)/F)<>((Z-1)/F)THEN470
430         IFBB$<>""THEN470
440         IFB$(1,3)="1" AND F>1
                PRINT@128,B$(2,1);:
                PRINT@T+64,Z$;
450         IFF>1PRINT@T+64,Z$;
460         GOTO390
470         D$(Z)=BB$:
            Z=Z+1:
        NEXTY:
    NEXTX
480 II=II+1: NR=NR+1: GOTO330
490 RETURN
495 '---LIST DATA ROUTINE----------------------------------------
500 XX=0: Z=1: IF NR<1 RETURN
510 XX=XX+1: CLS:
    IFAT$<>"AUTO ON"
       A$="*LIST* <E>-ESCAPE, <[>-BACKSPACE, <SHIFT>: <L>-LAST,
 <S>-SKIP"
    ELSE
       A$="LIST ROUTINE - PRESS <E> TO EXIT"
520 GOSUB1310 : GOSUB1140
530 IFAT$="AUTO ON":
       Z=Z-F:
       GOSUB1370
535 '---NOTE: THE CHARACTER TESTED IN ZZ$ IN LINES
536 '---540, 550, 580, 590, AND 600 IS SHIFTED
537 '---(I.E. PRESS THE <SHIFT> KEY, THEN THE CHARACTER).
538 '---TEST TO SEE IF SPECIAL FUNCTION IS DESIRED
540 IFZZ$="d" AND LEFT$(D$(Z-F),1) <> CHR$(191):
       D$(Z-F)=CHR$(191)+LEFT$(D$(Z-F),40):
       NR=NR-1
550 IFZZ$="u":
       Z=Z-F:
       GOSUB1180
560 IFZZ$=CHR$(91) AND XX=1:
       Z=Z-F:
       XX=XX-1
570 IFZZ$=CHR$(91) AND XX>1:
       XX=XX-2:
       Z=Z-(2*F)
580 IFZZ$="s":
       XX=XX+9:
       Z=Z+(9*F)
590 IFZZ$="1":
       XX=II-1:
       Z=(XX*F)+1
600 IFZZ$="p":
       Z=Z-F:
       GOSUB1370
610 IFZZ$="E": RETURN
615 '---IF XX<II THERE ARE MORE RECORDS TO LIST
620 IFXX<II
       THEN510
    ELSE
       AT$="AUTO OFF":
       RETURN
625 '---SAVE DATA ROUTINE---------------------------------------
630 IFNR<1: RETURN
640 CLS: A$="SAVE DATA ROUTINE": GOSUB1330
650 IFZZ$="E": RETURN
660 PRINT: PRINT"OLD FILE NAME IS <";F$;">": PRINT
670 INPUT"ENTER NEW FILE NAME ";F$
680 OPEN"O",1,F$+"/TXT"
690 PRINT#1,NR;F;H
700 FORCC=1TOH:
       PRINT#1,
       CHR$(34);B$(CC,1);CHR$(34);
       CHR$(34);B$(CC,2);CHR$(34);
       CHR$(34);B$(CC,3);CHR$(34):
    NEXT
705 '---ZZ=TOTAL NUMBER OF FIELDS (INCLUDING DELETE RECORDS)
710 ZZ=(II*F):
    FORCC=1TOZZ
715 '---TEST FOR DELETE RECORDS & DON'T WRITE THEM TO DISK
720    IFLEFT$(D$(CC),1)=CHR$(191)
          THENCC=CC+(F-1):
          GOTO740
730    PRINT#1,CHR$(34);D$(CC);CHR$(34)
740 NEXT: CLOSE: RETURN
745 '---SEARCH ROUTINE------------------------------------------
750 IFNR<1: RETURN
760 CLS: A$="*SEARCH*   <E> TO ESCAPE, ANY KEY TO CONTINUE":
    GOSUB1310 : GOSUB1320
770 IFZZ$="E": RETURN
780 FF=1: JJ(1)=1: Q(1)=1: YB=1
785 '---PRINT HEADINGS
790 CLS:
    FORX=1TOHSTEP2:
       PRINTB$(X,1);:
       PRINTTAB(40)B$(X+1,1):
    NEXT
800 PRINTSTRING$(63,"."):
    A$="TO FIND RECORDS TO BE DELETED, PUSH <ENTER> FOR HEADING
# 1":
    GOSUB1310
810 PRINTSTRING$(63,".")
815 '---FOLLOWING LINE LIMITS # OF TEXT SEARCHES TO 5
820 IF FF>5:
    YB=FF:
    GOTO930
```

Listing 3 continues

✓See List of Advertisers on Page 563

Listing 3 continued

```
830 PRINT"SELECT SEARCH HEADING #";FF;: PRINTTAB(38);: LINEINPUT
SS$(FF)
835 '---TEST FOR BLANK ON HEADING #1
840 IFSS$(1)="":
          S$=B$(1,1):
          QQ$(1)=CHR$(191):
          GOTO920
845 '---TEST FOR BLANK ON ANY HEADING
850 IFSS$(FF)="":
          YB=FF:
          GOTO930
855 '---ADD TRAILING PERIODS TO HEADING ENTRY
860 S$=SS$(FF)+RIGHT$(B$,21-LEN(SS$(FF)))
870 Q(FF)=0:
      FORX=1TOH:
          Q(FF)=Q(FF)+VAL(B$(X,3))
880       IFB$(X,1)=S$:
              PRINT"ENTER ";SS$(FF);" SOUGHT   ";:
              PRINTTAB(40);:
              LINEINPUTQQ$(FF):
              JJ(FF)=VAL(B$(X,3)):
              FF=FF+1:
              GOTO820
890 NEXT
900 A$="HEADING DOES NOT EXIST": GOSUB1330
910 IFZZ$="E"
          THENRETURN
      ELSE
          820
920 CLS: A$="SEARCHING FOR RECORDS TO BE DELETED": GOSUB1310 : G
OTO940
930 FF=1: CLS: A$="SEARCHING FOR A MATCH": GOSUB1310
940 FORX=1TOII*FSTEPF
945 '---Q(FF)=LAST RELATIVE FIELD POSITION OF HEADING FF
946 '---JJ(FF)= # OF FIELDS ATTACHED TO HEADING FF
947 '---G=FIRST FIELD OF HEADING IN CURRENT RECORD
948 '---E=LAST FIELD OF HEADING IN CURRENT RECORD
950     G=(X+Q(FF)-JJ(FF)): E=(G+JJ(FF)-1)
960     FORY=GTOE
970         L(FF)=INSTR(D$(Y),QQ$(FF))
980         IF L(FF)<>0:
                FF=FF+1:
                GOTO1030
990     NEXTY:
        PRINT@55,INT((X+F)/F):
        FF=1
1000 NEXTX: AT$="AUTO OFF"
1005 '---IF YOU GET HERE, THE RECORD SOUGHT DOESN'T EXIST
1010 CLS: PRINT: A$="NO MORE DATA TO SEARCH": GOSUB1330
1020 IFZZ$="E"
         THEN RETURN
     ELSE
         780
1025 '---IF FF<YB, THERE ARE MORE HEADINGS TO SEARCH IN RECORD
1030 IF FF<YB
         THEN950
     ELSE
         FF=1
1040 CLS: A$=" * FOUND RECORD *   PRESS <E> TO ESCAPE": GOSUB131
0
1050 Z=X: GOSUB1140
1060 IFAT$="AUTO ON":
         Z=Z-F:
         GOSUB1370
1065 '---NOTE: THE CHARACTER TESTED IN ZZ$ IN LINES
1066 '---1070, 1080, AND 1090 IS SHIFTED
1067 '---(I.E. PRESS THE <SHIFT> KEY, THEN THE CHARACTER).
1070 IFZZ$="d" AND LEFT$(D$(Z-F),1)<>CHR$(191):
         D$(Z-F)=CHR$(191)+D$(Z-F):
         Z=Z-F:
         NR=NR-1:
         GOTO1040
1080 IFZZ$="u":
         Z=Z-F:
         GOSUB1180 :
         Z=Z+F
1090 IFZZ$="p":
         Z=Z-F:
         GOSUB1370
1100 IFZZ$="E": RETURN
1105 '---IF YOU GET HERE, SEARCH FILE FOR ANOTHER MATCH
1110 CLS: A$="SEARCHING FOR A MATCH": GOSUB1310 : GOTO1000
1115 '---EXIT PROGRAM ROUTINE-----------------------------------
1120 CLS: PRINT: PRINT"VALID RECORDS -> ";NR: PRINT:
     A$="EXIT ROUTINE - DATA WILL BE LOST IF NOT SAVED BEFORE EX
IT": GOSUB1310
1130 PRINT:
     A$="PRESS:  <M>-MENU  OR  <E>-EXIT":
     GOSUB1310 : GOSUB1320 :
     IFZZ$<>"E"
         THENRETURN
     ELSE
         END
1135 '---DISPLAY DATA ROUTINE-----------------------------------
1140 GOSUB1340
1150 IFAT$<>"AUTO ON"
         PRINT"ANY KEY TO CONTINUE, <SHIFT>: <U>-UPDATE, <P>-PRI
NT, <D>-DELETE";:
     ELSE
         PRINTTAB(21)"AUTO PRINT FUNCTION ON";
1160 IFAT$="AUTO ON": RETURN
1170 GOSUB1320 : RETURN
1175 '---UPDATE ROUTINE-----------------------------------------
1180 CLS:
     A$="*UPDATE*  CHANGE, THEN <ENTER> ELSE <ENTER> TO ADVANCE
CURSOR": GOSUB1310
1185 '---DF=1 MEANS DELETE FLAG IS ON (RECORD MARKED TO DELETE)
1190 IFLEFT$(D$(Z),1)=CHR$(191)
         THENDF=1
     ELSE
         DF=0
1200 GOSUB1340
1210 PRINTSTRING$(63,"-");
1215 '---Z=Z-F MEANS BACK UP ONE RECORD
1220 Z=Z-F: T=22
1225 '---IF USER ENTERS DATA, PLACE IN D$(Z)
1230 FORW=1TOH:
         PRINT@T+42,B$(W,1);
1240     FORJ=1TOVAL(B$(W,3)):
             T=T+64
1250         PRINT@T+1,D$(Z);:PRINT@T,"";
1260         LINEINPUTBB$:
```

Listing 3 continues

Listing 3 continued

```
         BB$=LEFT$(BB$,41)
1270     IFBB$<>""THEND$(Z)=BB$
1280     PRINT@T+1,STRING$(41," ");:
         PRINT@T,D$(Z):
         Z=Z+1:
      NEXTJ:
   NEXTW
1285 '---IF RECORD WAS, BUT IS NOT NOW, MARKED TO DELETE
1286 '---ADD 1 TO NUMBER OF RECORDS (NR)
1290 IFDF=1 AND LEFT$(D$(Z-F),1)<>CHR$(191): NR=NR+1
1300 RETURN
1305 '---CENTER AND DISPLAY (A$) SUBROUTINE-------------------
1310 TB=.5 * LEN(A$): TB=32-TB: PRINTTAB(TB)A$: RETURN
1315 '---INKEY$ SUBROUTINE-----------------------------------
1320 ZZ$=INKEY$:
     IFZZ$=""
        THEN 1320
     ELSE
        RETURN
1325 '---DISPLAY, PROMPT, & INKEY$ SUBROUTINE-----------------
1330 GOSUB1310 : PRINT: A$="PRESS <E> TO ESCAPE, ANY KEY TO CONT
INUE":
     GOSUB1310 : GOSUB1320 : RETURN
1335 '---DISPLAY RECORD SUBROUTINE----------------------------
1340 FORV=1TOH:
        PRINTB$(V,1);
1350    FORU=1TOVAL(B$(V,3)):
           PRINTTAB(22)D$(Z):
           Z=Z+1:
        NEXTU:
     NEXTV
1360 RETURN
1365 '---PRINT RECORD SUBROUTINE------------------------------
1370 LN=66
1380 IF PEEK(14312)>127
       IF AT$="AUTO ON"
        IF PEEK(14337)<>32
         GOTO 1380 :
        ELSE
         AT$="AUTO OFF":
         ZZ$="E":
         Z=Z+F:
         RETURN:
       ELSE
        Z=Z+F:
        RETURN
1390 IFPEEK(16425)+F>LN-1: LPRINTCHR$(12);: POKE16425,1
1400 IFPEEK(16425)=1: LPRINT" "
1405 '---PRINT RECORD
1410 FORV=1TOH:
        LPRINTB$(V,1);
1420    FORU=1TOVAL(B$(V,3)):
           LPRINTTAB(22)D$(Z):
           Z=Z+1:
        NEXTU:
     NEXTV
1430 LPRINTSTRING$(63,"-")
1440 RETURN
1445 '---ENABLE AUTO PRINT SUBROUTINE------------------------
1450 CLS: PRINTCHR$(23): PRINTTAB(6)"AUTO PRINT ENABLED":
     FORHH=1TO500: NEXT: AT$="AUTO ON": RETURN.
1460 END
```

Math Hangman

by Tim Knight

Help your children learn addition, subtraction, multiplication, and division with this fun game—and brush up your own math skills, too.

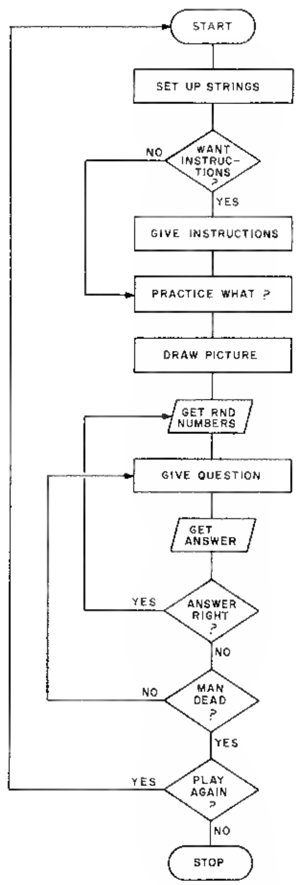


Fig. 1. Program Flowchart

Math Hangman is educational as well as entertaining. It helps people develop their skills in addition, subtraction, multiplication, and division, and it will probably be used mostly by young people. However, since many of us are becoming increasingly dependent upon calculators and computers, our mathematical skills are, no doubt, rusty. So, adults, too, may find this game challenging.

In Math Hangman, you answer questions about random math problems. For each wrong answer, a piece of the hangman appears. If the whole man is formed, you lose; if you get through all the problems, you win!

Extra Features

Just for fun, I put some extra features in this simple game. First, the playing instructions are in the game. Even though you might not need them, they are helpful to children and players who didn't type in the program.

Second, the computer repeats a problem if you get it wrong, providing a true test of your skills.

A third feature is found at the end of the game when the percentage for the number of right answers is given. This lets a player know if his skills are improving—or getting worse!

In case you want to modify the pro-

The Key Box

Models I & III
16K, 32K
Cassette or Disk Basic

IMPORTANT VARIABLE LIST

A	CHR$ DATA
B$(I)	PARTS OF MAN'S BODY
B (I)	PLACES ON SCREEN PIECES OF BODY GO
R1, R2	RANDOM NUMBERS
GU, GU$	PLAYER'S GUESS
AN, AN$	TRUE ANSWER
RT	RIGHT ANSWERS
WR	WRONG ANSWERS
P	PERCENTAGE

Table 1. Variables

gram, I've included a flowchart and a list of important variables. You might want to include easier problems, or ones that are more difficult. This program was not designed solely as an instruction course; it can also serve as a skeleton for other programmers who want to write educational programs. ■

Tim Knight is a 16-year-old high school student. He can be reached at 10 Fieldbrook Place, Moraga, CA 94556.

Program Listing

```
10 '    Math Hangman program, by Tim Knight
20 '    10 Fieldbrook Place
30 '    Moraga, CA   94556
40 '    for educational use on a TRS-80 Model I or III
50 CLEAR 500
60 CLS
70 DEFSNG A-Z
80 FOR I=1 TO 6
90 READ A
100 IF A=999 THEN NEXT ELSE IF A=333 THEN 190 ELSE IF A<128 AND
A<>26 AND A<>32 AND A<>24 THEN READ B: FOR J=1 TO A-1: B$(I)=B$(
I)+CHR$(B): NEXT: GOTO 90
110 B$(I)=B$(I)+CHR$(A)
120 GOTO 90
130 DATA 176, 186, 3, 176, 26, 7, 24, 188, 2, 191, 187, 143, 183
, 3, 191, 188, 26, 8, 24, 131, 143, 191, 2, 179, 191, 143, 131,
999
140 DATA 2, 188, 7, 191, 2, 188, 26, 9, 24, 9, 191, 26, 9, 24, 9
, 191, 999
150 DATA 3, 176, 2, 188, 7, 191, 2, 188, 26, 11, 24, 2, 191, 26,
 2, 24, 2, 191, 999
160 DATA 2, 188, 7, 191, 2, 188, 3, 176, 26, 2, 24, 2, 191, 26,
2, 24, 2, 191, 999
170 DATA 2, 191, 26, 2, 24, 2, 191, 26, 4, 24, 3, 188, 2, 191, 9
99
180 DATA 2, 191, 26, 2, 24, 2, 191, 26, 2, 24, 2, 191, 3, 188, 3
33
190 ' PROGRAM BEGINS HERE
200 B(1)=210
210 B(2)=401
220 B(3)=399
230 B(4)=401
240 B(5)=595
250 B(6)=598
260 PRINT TAB(20)"= MATH HANGMAN ="
270 PRINT
280 INPUT"Would you like instructions ";RE$
290 IF LEFT$(RE$, 1)="Y" THEN CLS ELSE GOTO 470
300 PRINT TAB(15)"* MATH HANGMAN INSTRUCTIONS *"
310 PRINT
320 PRINT"  This game is very simple to use.  It will both teach
 a person"
330 PRINT"and also serve as entertainment!  Its main purpose, th
ough,"
340 PRINT"is to help a person with his or her math skills."
350 PRINT"  The program will begin by asking if you would like t
o work"
360 PRINT"on addition, subtraction, multiplication, or division.
  You"
370 PRINT"chose one of these that you want most.  After that, ty
pe in "
380 PRINT"how many problems you would like to do, and the comput
er will"
390 PRINT"take it from there!"
400 PRINT"  You will do problems from then on.  For each one you
 get wrong";
410 PRINT"though, another piece of the hangman will come up.  If
 the"
420 PRINT"whole hangman is formed, then you lose.  However, if y
ou make"
430 PRINT"it through those problems without any trouble, you win
!"
440 PRINT
450 PRINT TAB(10)"Hit the space bar to begin game....";
460 IF PEEK(14400)=128 THEN CLS ELSE 460
470 CLS
480 PRINT
490 PRINT
500 PRINT"WOULD YOU LIKE TO PRACTICE ADDITION, SUBTRACTION, MULT
IPLICATION";
510 PRINT"OR DIVISION? (TYPE IN A,S,M, OR D) ";
520 INPUT CH$
530 CH$=LEFT$(CH$, 1)
```

Listing continues

Listing continued

```
54Ø IF CH$<>"A" AND CH$<>"S" AND CH$<>"M" AND CH$<>"D" THEN PRIN
T: PRINT"TRY AGAIN!": FOR K=1 TO 1ØØØ: NEXT: GOTO 47Ø
55Ø INPUT"AND HOW MANY PROBLEMS WOULD YOU LIKE TO TRY";NP
56Ø IF NP>5Ø THEN PRINT"WOW!  I THINK THAT'S A LITTLE TOO MUCH!"
: PRINT: GOTO 55Ø
57Ø IF NP<7 THEN PRINT"AWW, YOU CAN GIVE IT A BETTER SHOT THAN T
HAT!": PRINT: GOTO 55Ø
58Ø CLS
59Ø PRINT @ 832, STRING$(36, 191);
6ØØ PRINT @ 896, STRING$(3, 191);"          MATH HANGMAN
";STRING$(3, 191)
61Ø PRINT @ 784, CHR$(131)+CHR$(191)+STRING$(6, 131)+CHR$(191)+C
HR$(131);
62Ø PRINT @ 96Ø, STRING$(36, 191);
63Ø FOR Y=136 TO 776 STEP 64
64Ø PRINT @ Y, CHR$(191);
65Ø NEXT
66Ø PRINT @ 137, STRING$(1Ø, 131); CHR$(191);
67Ø PRINT @ 29Ø,"WHAT IS :";
68Ø R1=RND(1ØØ)
69Ø R2=RND(1ØØ)
7ØØ IF CH$="A" THEN C$="+": AN=R1+R2 ELSE IF CH$="S" THEN C$="-"
: AN=R1-R2 ELSE IF CH$="M" THEN C$="*": AN=R1*R2 ELSE C$="/": AN
=R1/R2
71Ø PRINT @ 354, R1; C$; R2;
72Ø INPUT GU
73Ø POKE 1536Ø+384+8, 191
74Ø PRINT @ 29Ø,"                    ";
75Ø PRINT @ 354,"                  ";
76Ø GU$=STR$(GU)
77Ø AN$=STR$(AN)
78Ø IF LEFT$(GU$, 5)=LEFT$(AN$, 5)THEN PRINT @ 29Ø,"R I G H T !"
;: FOR XX=WR TO 1 STEP -1: PRINT @ B(XX), B$(XX);: NEXT: RT=RT+1
: FOR K=1 TO 1ØØØ: NEXT: PRINT @ 29Ø,"           ";:IF RT+WR<NP
THEN 67Ø ELSE 85Ø
79Ø IF LEFT$(GU$, 5)<>LEFT$(AN$, 5)THEN WR=WR+1: FOR XX=WR TO 1
STEP -1: PRINT @ B(XX), B$(XX);: NEXT: PRINT @ 29Ø,"W-R-O-N-G";:
 FOR X=1 TO 1ØØØ: NEXT:IF WR>6 THEN 8ØØ ELSE PRINT @ 29Ø,"
     ";:IF RT+WR<NP THEN GOTO 71Ø ELSE 85Ø
8ØØ PRINT @ 784,"           ";
81Ø PRINT @ 283,"A h h h h h h h h h h h !";
82Ø FOR X=1 TO 1ØØØ
83Ø NEXT
84Ø CLS
85Ø CLS
86Ø PRINT"YOU HAD ";RT;"RIGHT AND";WR;"WRONG.  THAT COMES TO"
87Ø P=RT/(RT+WR)
88Ø P=P*1ØØ
89Ø PRINT P;"% RIGHT.  WOULD YOU LIKE TO TRY AGAIN ?";
9ØØ Q$=INKEY$
91Ø A$=INKEY$
92Ø IF A$="" THEN 91Ø
93Ø IF A$="Y" THEN RUN
94Ø END
```

Smartcat

by Irwin Rappaport

The Lynx modem and Emterm software make an impressive communications package; Smartcat makes this combination even more powerful.

The program Emterm, supplied with the Lynx modem (Emtrol Systems Inc., Lancaster, PA), creates a smarter TRS-80 terminal. Emterm and Lynx provide telephone dialing from the keyboard, storage and subsequent operator-commanded transmission of a message, alteration of certain operating parameters under software control, and simultaneous screen and line-printer (parallel) output. Emterm can also output control-key codes required by some data banks, and provides for up to 10 user-defined control keys.

These are powerful capabilities, but your TRS-80 terminal can do much more. You can discard Emterm and buy a more extensive program that supports the Lynx, or you can add the following features to Emterm:

- Uploading and downloading (transmitting and receiving) of program files.
- Support for upper/lowercase modification.
- Screen dump to line printer.
- Delay routine for keyboard debounce or transmission rate adjustments.
- Telephone directory storage with instant phone-number retrieval.

The Key Box

Disk Basic
Assembly Language
Model I or III
32K RAM
One disk drive
Editor/Assembler
Lynx modem

Passing Control to Smartcat

Three program listings are presented here. The two Basic programs play supporting roles while Smartcat, the Assembly listing, is the addition to Emterm. When Emterm is loaded and executed, it disassociates itself from the TRS-80 operating system and runs its own keyboard scan routine, enabling output of control-key codes. The first of two jump-to-Smartcat patches is made at this scan routine, location 7BB8 hex in Emterm.

While Emterm is running, frequent calls are made to 7BB8H to determine keyboard status and to decode the matrix if a key is pressed. The ASCII value of the key is sent to the accumulator. By having control at that point, Smartcat can test the accumulator and take whatever action has been programmed for a particular key. For example, the program contains a routine to dump the screen contents to a line printer when you press the "zero" key. Smartcat checks the accumulator and branches to the dump if it finds 30 hex.

In effect, the Lynx/TRS-80 gains the ability to learn whatever you teach it. A terminal must know whether its user wants a program routine or the actual character when a key is pressed. Emterm makes this distinction by incorporating the down arrow as a control key while Smartcat requires that shift, @ be used as the control-key sequence.

Since Emterm resides from 7537H–7E03H and uses all the area below it through 7100H for the stack and message storage, the Smartcat Assembly listing (see Program Listing 1) begins at 7F00H. The two programs are combined and saved in one package, from 7537H–880FH with the entrance at 7F00H. Upon execution, the Smartcat code creates two patches into Emterm: one for the keyboard-scan jump routine and the other to make provision for downloading. In addition, a patch is made into TRSDOS at 401EH for lowercase display support.

The lowercase driver routine supports the 2102 RAM chip piggyback modification, such as the kit marketed by Emanuel B. Garcia, Jr. and Associates, Chicago, IL. In Listing 1, the location labeled SWLC is specifically included for those not having the lowercase modification. If you do not have the lowercase modification, change the instruction from RET (C9) to NOP (00), which negates the lowercase routine. If you install the modification at some later date, change it back.

Emterm jumps to Smartcat at

UTENTR. The first three bytes there are the instructions originally contained in 7BB8,9,AH, and which were replaced with the jump-out patch. Then the remainder of the keyboard scan is called. Regaining control upon the return, Smartcat calls SWD, the delay routine.

SWD initially contains an RET (C9) instruction. When the delay routine is activated by issuing the appropriate keyboard command, location SWD is changed to NOP (00), thereby allowing the call to the delay routine to proceed. The contents of the BC register pair determine the amount of delay. Three levels of delay are user selectable from the keyboard as determined in FUNC10, FUNC11, and FUNC12 at the LD instruction (02,10,80 hex respectively). You may vary all three values.

The call to FUNC begins the routine that permits Smartcat to distinguish a program function command from an ordinary keyboard character when a key is pressed. If FUNC finds a shift, @ (60H) in the accumulator, switch SWF is changed from RET to NOP, similarly to the action at the SWD switch. When FUNC is called upon the next key press, the routine reaches SWF, and since it contains 00, passes through it. The switch is then closed by reloading it with C9, and the routine continues to search for the commanded function.

"(Emterm has) powerful capabilities, but your TRS-80... can do much more."

Switches SWK and SWU are called next. SWK opens when either a phone number or an upload transmission is wanted. For keyboard shift reversing (shift for capitals, no shift for lowercase), SWU is open. Both switches are toggled with one of the keyboard function command routines. After completing all the Smartcat calls, program action returns to the point in Emterm where the keyboard scan was originally called.

Upload, Download, and the Telephone Directory

For uploading, the file to be transmitted is placed into a RAM buffer area beginning with two control bytes at 88FEH and the actual file following at 8900H. The Basic program UPDOWN/CAT (see Program Listing 2) loads the buffer. Then when in Emterm terminal mode, a keyboard command begins the upload transmission.

To download a file, the same buffer area is used. Upon keyboard command, Smartcat begins loading every byte received by the Lynx into the RAM area starting at 8900H. Downloading continues until you issue a stop command from the keyboard, or the designated end of the buffer is reached. If the downloaded file was an ASCII format Basic program, it would be subsequently taken from RAM and saved

Program Listing 1. Smartcat

```
                 ;*** SMARTCAT *** OMIT LOWER CASE PORTION FOR MODEL III
7FØØ                      ORG     7FØØH
7BB8             CMDEMT   EQU     7BB8H   ;JUMP OUT POINT FROM EMTERM KEY-SCAN

81FF             TABLE    EQU     81FFH   ;START PHONE NUMBER TABLE
85ØØ             DIRECT   EQU     85ØØH   ;START DIRECTORY
88FE             BUFFER   EQU     88FEH   ;BEGINNING OF UPLOAD BUFFER
7FØØ 3EC3                 LD      A,ØC3H
7FØ2 325779               LD      (7957H),A;EMTERM JUMP-OUT TO DNLOAD
7FØ5 21CF8Ø               LD      HL,DNLOAD
7FØ8 225879               LD      (7958H),HL
7FØB 211D7F               LD      HL,LCENTR        ;LOWER CASE DRIVER PATCH ...

7FØE 221E4Ø               LD      (4Ø1EH),HL       ;OMIT FOR MODEL III
7F11 32B87B               LD      (CMDEMT),A
7F14 21447F               LD      HL,UTENTR
7F17 22B97B               LD      (CMDEMT+1),HL
7F1A C33775               JP      7537H   ;ALSO TRY USING 755C ... SEE TEXT
7F1D CD3E7F      LCENTR   CALL    SWLC    ;LOWER CASE DRIVER BEGINS ...
7F2Ø DD6EØ3               LD      L,(IX+Ø3);OMIT DRIVER ROUTINE FOR MOD III
7F23 DD66Ø4               LD      H,(IX+Ø4)
7F26 DA9AØ4               JP      C,Ø49AH
7F29 DD7EØ5               LD      A,(IX+Ø5)
7F2C B7                   OR      A
7F2D 28Ø1                 JR      Z,ENTR1
7F2F 77                   LD      (HL),A
7F3Ø 79          ENTR1    LD      A,C
7F31 FE2Ø                 CP      2ØH
7F33 DAØ6Ø5               JP      C,Ø5Ø6H
7F36 FE8Ø                 CP      8ØH
7F38 D2A6Ø4               JP      NC,Ø4A6H
7F3B C37DØ4               JP      Ø47DH
7F3E C9          SWLC     RET
7F3F 33                   INC     SP
7F4Ø 33                   INC     SP
7F41 C358Ø4               JP      Ø458H   ;END OF LOWER CASE DRIVER ROUTINE
7F44 D9          UTENTR   EXX             ;ORIGINAL EMTERM CODE
7F45 3EFF                 LD      A,ØFFH  ;ORIGINAL EMTERM CODE
7F47 CDBB7B               CALL    CMDEMT+3
7F4A CD6F8Ø               CALL    SWD
7F4D B7                   OR      A
7F4E C4587F               CALL    NZ,FUNC
7F51 CD868Ø               CALL    SWK
7F54 CD5B8Ø               CALL    SWU
7F57 C9                   RET
7F58 FE6Ø        FUNC     CP      6ØH
7F5A 2ØØ6                 JR      NZ,SWF
7F5C F1                   POP     AF
7F5D AF                   XOR     A
7F5E 32627F               LD      (SWF),A
7F61 C9                   RET
7F62 C9          SWF      RET
7F63 F5                   PUSH    AF
7F64 3EC9                 LD      A,ØC9H
7F66 32627F               LD      (SWF),A
7F69 F1                   POP     AF
7F6A FE3A        FUNC1    CP      3AH     ;KEY<:>=UPPER/LOWER REVERSE
7F6C 2ØØ6                 JR      NZ,FUNC2
7F6E F1                   POP     AF
7F6F AF                   XOR     A
7F7Ø 325B8Ø               LD      (SWU),A
7F73 C9                   RET
7F74 FE2A        FUNC2    CP      2AH     ;KEY<*>=REVERSE OFF
7F76 2ØØ8                 JR      NZ,FUNC3
7F78 F1                   POP     AF
7F79 3EC9                 LD      A,ØC9H
7F7B 325B8Ø               LD      (SWU),A
7F7E AF                   XOR     A
7F7F C9                   RET
7F8Ø FE3B        FUNC3    CP      3BH     ;KEY<;>=DELAY
7F82 2ØØ6                 JR      NZ,FUNC4
7F84 F1                   POP     AF
7F85 AF                   XOR     A
7F86 326F8Ø               LD      (SWD),A
7F89 C9                   RET
7F8A FE2B        FUNC4    CP      2BH     ;KEY<+>=DELAY OFF
7F8C 2ØØ8                 JR      NZ,FUNC5
7F8E F1                   POP     AF
7F8F 3EC9                 LD      A,ØC9H
7F91 326F8Ø               LD      (SWD),A
7F94 AF                   XOR     A
7F95 C9                   RET
7F96 FE3Ø        FUNC5    CP      3ØH     ;KEY<Ø>=PRINT SCREEN
```

Listing continues

Listing continued

```
7F98 2025             JR      NZ,FUNC6
7F9A E5               PUSH    HL
7F9B 21003C           LD      HL,3C00H
7F9E CD2080     AGAIN CALL    READY
7FA1 7E               LD      A,(HL)
7FA2 32E837           LD      (37E8H),A
7FA5 23               INC     HL
7FA6 7D               LD      A,L
7FA7 CBBF             RES     7,A
7FA9 CBB7             RES     6,A
7FAB B7               OR      A
7FAC 20F0             JR      NZ,AGAIN
7FAE CD2080           CALL    READY
7FB1 3E0D             LD      A,0DH
7FB3 32E837           LD      (37E8H),A
7FB6 7C               LD      A,H
7FB7 FE40             CP      40H
7FB9 20E3             JR      NZ,AGAIN
7FBB E1               POP     HL
7FBC F1               POP     AF
7FBD AF               XOR     A
7FBE C9               RET
7FBF FE40       FUNC6 CP      40H      ;KEY<@>=SHOW PHONE DIRECTORY
7FC1 2013             JR      NZ,FUNC7
7FC3 E5               PUSH    HL
7FC4 D5               PUSH    DE
7FC5 C5               PUSH    BC
7FC6 210085           LD      HL,DIRECT
7FC9 11803C           LD      DE,3C80H
7FCC 010003           LD      BC,0300H
7FCF EDB0             LDIR
7FD1 C1               POP     BC
7FD2 D1               POP     DE
7FD3 E1               POP     HL
7FD4 AF               XOR     A
7FD5 C9               RET
7FD6 FE2F       FUNC7 CP      2FH      ;KEY</>=DOWNLOAD ON
7FD8 2008             JR      NZ,FUNC8
7FDA AF               XOR     A
7FDB 32D180           LD      (SWDN),A
7FDE CD2880           CALL    ONMSG
7FE1 C9               RET
7FE2 FE3F       FUNC8 CP      3FH      ;KEY<?>=DOWNLOAD OFF
7FE4 200A             JR      NZ,FUNC9
7FE6 3EC9             LD      A,0C9H
7FE8 32D180           LD      (SWDN),A
7FEB CD2E80           CALL    OFFMSG
7FEE AF               XOR     A
7FEF C9               RET
7FF0 FE2E       FUNC9 CP      2EH      ;KEY<.>=BEGIN UPLOAD TRANSMISSION
7FF2 200A             JR      NZ,FUNC10
7FF4 E5               PUSH    HL
7FF5 21FE88           LD      HL,BUFFER;1ST BYTE FF, THEN KEY<.>,THEN PROG

7FF8 22EF80           LD      (WDADRS),HL      ;CHANGE ADR TO PROG.
7FFB E1               POP     HL
7FFC 181B             JR      FUNC20   ;GO TO TRANSMIT
7FFE FE31      FUNC10 CP      31H      ;KEY<1>=DELAY AMOUNT OF 02
8000 2004             JR      NZ,FUNC11
8002 3E02             LD      A,02
8004 180E             JR      AMTLD
8006 FE32      FUNC11 CP      32H      ;KEY<2>=DELAY AMOUNT OF 10H
8008 2004             JR      NZ,FUNC12
800A 3E10             LD      A,10H
800C 1806             JR      AMTLD
800E FE33      FUNC12 CP      33H      ;KEY<3>=DELAY AMOUNT OF 80H
8010 2007             JR      NZ,FUNC20
8012 3E80             LD      A,80H
8014 327480     AMTLD LD      (AMTDEL+1),A
8017 AF               XOR     A
8018 C9               RET
8019 F5        FUNC20 PUSH    AF       ;WILL OUTPUT PHONE NUMBERS/UPLOAD
                                        OR SWK WILL CLOSE IF NEITHER
801A AF               XOR     A
801B 328680           LD      (SWK),A
801E F1               POP     AF
801F C9               RET
8020 3AE837     READY LD      A,(37E8H)        ;CHECK PRINTER STATUS
8023 CB7F             BIT     7,A
8025 C8               RET     Z
8026 18F8             JR      READY
8028 E5         ONMSG PUSH    HL       ;PRINT DOWNLOAD ON MESSAGE TO SCREEN

8029 214080           LD      HL,DOWNON
802C 1804             JR      PRTMSG
802E E5        OFFMSG PUSH    HL       ;DOWNLOAD OFF MESSAGE
802F 214C80           LD      HL,DOWNOF
8032 D5        PRTMSG PUSH    DE       ;BLOCK TRANSFER TO SCREEN
8033 C5               PUSH    BC
8034 11C03F           LD      DE,3FC0H
8037 010C00           LD      BC,000CH
803A EDB0             LDIR
803C C1               POP     BC
803D D1               POP     DE
803E E1               POP     HL
803F C9               RET
8040 44        DOWNON DEFM    'DOWNLOAD ON '
804C 44        DOWNOF DEFM    'DOWNLOAD OFF'
8058 00               NOP
8059 00               NOP
805A 00               NOP
805B C9           SWU RET
805C FE41             CP      41H
```

Listing continues

on disk using UPDOWN/CAT from Basic.

Smartcat provides for storage of 55 telephone numbers, any of which may be called with a keyboard command. In addition, a directory is included so the user can easily determine which phone number belongs to whom, and what key brings it up. You use the directory and phone-number functions at the point in Emterm where you would ordinarily enter the number to be dialed from the keyboard. With the Basic program PHONEDIR/CAT (see Program Listing 3) you can establish, review, and revise the directory. Table, at the beginning of Listing 1, sets the start of the phone numbers at 81FFH. Similarly, Direct dictates that the informational portion begins at 8500H.

Creating the Programs

Assemble Smartcat with an assembler, or load the object code directly with Debug. You may omit all NOPs in the listing.

Using Debug, load location 81FFH with FFH. Again with Debug, load all locations from 8200H–880FH with 20H. Then load location 84FFH with 00H.

Load but do not execute Emterm. Be sure all the above, including Smartcat, is in memory. Save the entire package on disk using Dump or Tapedisk. Pick a suitable name for the combined program which starts at 7537H, ends at 880FH, and has its entrance at 7F00H. Then type in and save the two Basic programs, PHONEDIR/CAT and UPDOWN/CAT.

Putting the Programs to Work

To set up a telephone directory, load the combined Smartcat/Emterm program, but do not execute it. Enter Basic, enter 0 for files, reserve memory at 30006, and run PHONEDIR/CAT. Press enter at the prompt to add new listings. Input as prompted, first any key desired, then the name of the party (10-character limit), and finally the phone number. Ten digits are available for the number so you can use area codes where needed.

Enter telephone numbers just as you would for Emterm dialing using digits only, no punctuation. Continue with a few more entries, and end by pressing enter instead of a new key. The entries will be POKEd into Smartcat and you will be reminded to save the Smartcat/Emterm combination, now containing your new directory.

Before doing that, run the Basic program again but this time enter D to see the directory. Page 1 scrolls onto the

screen. Wait for the page ending prompt. PHONEDIR/CAT displays three directory pages, each ending with the same prompt. Pressing enter produces the next page while pressing G gets back to the three choices at the beginning of the program. Input G and then try C to change a listing. Input any previously entered directory key, then the revised name and number at the appropriate prompts. Review the directory to be sure that the change has been made. Save Smartcat/Emterm each time after revising the directory.

Uploading preparation begins in Basic. Run UPDOWN/CAT and press 2 for uploading. Enter the file name (the file should be a Basic program in ASCII format) and unless you have made some program revisions, press enter at the starting-address prompt that appears after the disk drive stops. The file is printed to the screen and POKEd into RAM, sector by sector. Enter CMD"S" and load and execute Smartcat/Emterm in the usual way. When you are ready to transmit the file, press shift, @ and then the period key. Uploading will proceed, returning to Emterm terminal mode upon completion. The receiving party must process your transmission according to his own system's needs.

You may transmit anything contained in the RAM buffer with the upload routine. If you will be sending a non-Basic file, place it in the buffer in any way desired—load it, block transfer it, or type it in, beginning at 8900H. At 88FEH and 88FFH put the start codes FFH and 3EH respectively (UPDOWN/CAT puts them in when loading a Basic file). Then upload from Emterm as before.

To download any Basic file, or anything your Lynx receives, press shift, @ and then the slash key. "DOWNLOAD ON" will appear on your screen, and every byte received thereafter will be stored in RAM beginning at 8900H. Downloading continues until you press shift, @ followed by the question-mark key and "DOWNLOAD OFF" appears (or the end of the buffer is reached). If desired, you may download another file immediately using the same procedure. It will be stored in the buffer following the previous file.

You can save received files on disk when you terminate the phone connection. If the downloaded file is an ASCII-format Basic program that you will put on disk with UPDOWN/CAT, follow this procedure after hanging up the phone:

Listing continued

```
805E F8              RET    M
805F FE61            CP     61H
8061 3006            JR     NC,UL1
8063 FE5B            CP     5BH
8065 D0              RET    NC
8066 C620            ADD    A,20H
8068 C9              RET
8069 D620    UL1     SUB    20H
806B C9              RET
806C 00              NOP
806D 00              NOP
806E 00              NOP
806F C9      SWD     RET
8070 E5              PUSH   HL
8071 C5              PUSH   BC
8072 F5              PUSH   AF
8073 0602    AMTDEL  LD     B,02H    ;AMOUNT OF DELAY
8075 0E01            LD     C,01
8077 AF              XOR    A
8078 0B      MORDEL  DEC    BC
8079 B8              CP     B
807A 20FC            JR     NZ,MORDEL
807C F1              POP    AF
807D C1      OUTDEL  POP    BC
807E E1              POP    HL
807F C9              RET
8080 00              NOP
8081 00              NOP
8082 00              NOP
8083 00              NOP
8084 00              NOP
8085 00              NOP
8086 C9      SWK     RET
8087 E5              PUSH   HL
8088 2AEF80          LD     HL,(WDADRS)
808B F5              PUSH   AF
808C 3EFF            LD     A,0FFH
808E BE              CP     (HL)
808F 2022            JR     NZ,MIDDLE
8091 F1              POP    AF
8092 B7              OR     A
8093 2002            JR     NZ,GOON
8095 E1              POP    HL
8096 C9              RET
8097 23      GOON    INC    HL
8098 BE      KEY1    CP     (HL)
8099 2810            JR     Z,KEY3
809B F5              PUSH   AF
809C 23      KEY2    INC    HL
809D 3EBF            LD     A,0BFH
809F BE              CP     (HL)
80A0 20FA            JR     NZ,KEY2
80A2 23              INC    HL
80A3 3EFF            LD     A,0FFH
80A5 BE              CP     (HL)
80A6 2818            JR     Z,KEY4
80A8 F1              POP    AF
80A9 18ED            JR     KEY1
80AB 23      KEY3    INC    HL
80AC 7E              LD     A,(HL)
80AD 23              INC    HL
80AE 22EF80          LD     (WDADRS),HL
80B1 E1              POP    HL
80B2 C9              RET
80B3 F1      MIDDLE  POP    AF
80B4 3EBF            LD     A,0BFH
80B6 BE              CP     (HL)
80B7 2808            JR     Z,KEY5
80B9 7E              LD     A,(HL)
80BA 23              INC    HL
80BB 22EF80          LD     (WDADRS),HL
80BE E1              POP    HL
80BF C9              RET
80C0 F1      KEY4    POP    AF
80C1 21FF81  KEY5    LD     HL,TABLE
80C4 22EF80          LD     (WDADRS),HL
80C7 E1              POP    HL
80C8 3EC9            LD     A,0C9H
80CA 328680          LD     (SWK),A
80CD AF              XOR    A
80CE C9              RET
80CF E67F    DNLOAD  AND    7FH      ;BEGINNING OF DOWN-LOAD ROUTINE
80D1 C9      SWDN    RET
80D2 E5              PUSH   HL
80D3 F5              PUSH   AF
80D4 2AED80          LD     HL,(BUFADR)
80D7 77              LD     (HL),A
80D8 23              INC    HL
80D9 22ED80          LD     (BUFADR),HL
80DC 3EE0            LD     A,0E0H   ;ESTABLISH END OF BUFFER AT E000H
80DE BC              CP     H
80DF 2803            JR     Z,BUFEND
80E1 F1              POP    AF
80E2 E1              POP    HL
80E3 C9              RET
80E4 F1      BUFEND  POP    AF
80E5 E1              POP    HL
80E6 3E3F            LD     A,3FH
80E8 C3E27F          JP     FUNC8
80EB E1              POP    HL
80EC C9              RET
80ED 0089    BUFADR  DEFW   BUFFER+2;INITIALIZE BUFFER ADR
80EF FF81    WDADRS  DEFW   TABLE    ;INITIALIZE TABLE FOR PHONE.
0000                 END
```

● Enter Debug and examine the area at 8900H.

● Determine the beginning address of the file and make a note of it for later use.

● Locate the last byte of the file and load 00 into the location immediately after it.

● Go into Basic and run UPDOWN/CAT. Enter 1 for the download save.

● Enter a file name for the program as desired.

● Enter the program starting address noted before (decimal required). The easiest way to obtain the decimal value is to press break, use &H to get the hex to decimal conversion, type and enter CONT, and input the decimal.

● The lines will be printed on screen as saving proceeds.

```
10 CLEAR 1000:CLS:PRINT:PRINT"GET FILES TO AND FROM DISK":PRINT:
PRINT"1 - SAVE DOWNLOADED FILE FROM MEMORY TO DISK":PRINT"2 - GE
T FILE FROM DISK TO MEMORY FOR UPLOAD TRANSMISSION":PRINT:PRINT"
ENTER LINE NUMBER FOR ROUTINE DESIRED"
20 R$=INKEY$:IF R$="" OR VAL(R$)>2 OR VAL(R$)=0 THEN 20 ELSE ON
VAL(R$) GOSUB 100,200:GOTO 10
100 REM *** START OF TO-DISK SEGMENT ***
110 CLS:PRINT"MEMORY TO DISK ROUTINE":LINEINPUT"FILE NAME: ";F$:
OPEN"O",1,F$
120 INPUT"START OF FILE IN MEMORY";P
130 PRINT#1,CHR$(PEEK(P));:PRINTCHR$(PEEK(P));:P=P+1:IFPEEK(P)=0
0THEN140   :ELSE130
140 RETURN
200 REM *** START OF FROM-DISK SEGMENT ***
210 CLS:PRINT"DISK TO MEMORY ROUTINE":LINEINPUT"FILE NAME: ";F$:
OPEN"I",1,F$
220 A=&H88FE:INPUT"PROGRAM WILL START AT 88FE.  <ENTER> IF OK OR
 ENTER DESIRED      STARTING ADDRESS";A:POKEA,&HFF:A=A+1:POKEA,&H
2E:A=A+1
230 LINEINPUT#1,A$:PRINTA$:FORX=1TOLEN(A$):POKEA,ASC(MID$(A$,X,1
)):A=A+1:IFA>&HE800THENPRINT"BUFFER FULL":GOTO240   :ELSENEXTX:PO
KEA,&H0D:A=A+1:IFEOF(1)THEN240   :ELSEGOTO230
240 POKEA,&HBF:A=A+1:POKEA,&HFF
250 RETURN
```

Program Listing 2. UPDOWN/CAT

If you are going to return to modem operation, start fresh by reloading and executing Smartcat/Emterm.

Using Smartcat's other features is a matter of pressing the right keys. To obtain the telephone directory, press shift, @ followed by the unshifted @ key. The entire directory, showing each party's name and the key that will bring up his phone number, will be block transferred to the screen. To have the phone number come up when in the Emterm dialing mode, press shift, @ and then the key wanted. Press enter as usual and the Emterm function dials the number just as if you had typed it in.

The screen-dump-to-line-printer feature is useful when you do not select the Emterm function for simultaneous video and printer output, but then decide you want a printout of the current screen contents. To print the screen, press shift, @ and then the zero key. Reversing the keyboard shift for upper/lowercase is accomplished with shift, @ once again, followed by the colon key. Shift, @ and the asterisk key returns the normal keyboard. Shift, @ and the semicolon key turns on the delay feature, while shift, @ and the plus-sign key turns it off. To select the desired amount of delay, press shift, @ and then either the one, two, or three key.

Model III Notes

For the Model III, eliminate the lines in Smartcat for lowercase display as indicated in the remarks. Uploading, downloading, the directory, and the delay feature work equally well with either the Model I or III.

If you wish to expand Smartcat, simply continue the series following FUNC12 so that a branch to your subroutine is produced at the press of shift, @ and a designated key. The accumulator is normally zeroed prior to the return from a FUNC. That action avoids printing or transmitting the character. In effect, Emterm will never know that a key was pressed.

You can also write routines to be activated upon receiving a specific character. As Emterm jumps to Smartcat at DNLOAD, the first three bytes found are the instructions originally contained in Emterm. The return at SWDN serves as an original instruction and the download switch. At that point the control is in Smartcat and the accumulator contains the last received byte, thereby providing the opportunity to devise your routine.

When executing Emterm, with or without Smartcat, try using an entrance address of 755CH instead of 7537H so the beginning of the program is bypassed. Do that by loading but not executing Emterm, then load and execute Debug, and finally enter the program with G 755CH. You can then leave Emterm and enter Debug with the break key, and return to Emterm by entering G from Debug. This will not terminate a phone connection if there was one prior to the break. To incorporate this into Smartcat, change the jump address in Listing 1 line 160 from 7537H to 755CH.

The final note involves somewhat of a mystery. The assembler listing for Smartcat places the end of the available download buffer considerably below the actual end of memory. It seems that Lynx causes some bit resets at certain locations in very high memory. If that problem doesn't appear in your system, then the buffer can end at the top of RAM. In any case, use only the most significant byte to designate the buffer end, as in assembler line 2710. ■

Irwin Rappaport can be reached at 24 Hemlock Hill Road, Upper Saddle River, NJ 07458.

```
5 REM *** PHONEDIR/CAT ***  NOTE THAT CLEAR STRING SPACE LIMITS
NUMBER OF NEW ENTRIES THAT MAY BE MADE AT ONE TIME
10 CLEAR1000:DIM K$(55),N$(55),P$(55):CLS
20 X=0:C$="":INPUT"<ENTER> TO ADD NEW LISTINGS ... OR ...
ENTER 'C ' TO CHANGE A LISTING ... OR ...
ENTER 'D ' TO SEE DIRECTORY";C$:IFC$="C"THEN110:ELSEIFC$="O"THEN
GOSUB500:GOTO20:ELSEIFC$<>""THEN20
50 P=&H8540
60 IFPEEK(P)=128THEN70:ELSEP=P+1:GOTO60
70 A=55-PEEK(&H84FF):PRINT"SPACE AVAILABLE FOR";A;"ADDITIONAL PH
ONE NUMBERS":S=PEEK(&H84FF)+1
100 FORX=STO55
110 INPUT"KEY";K$(X):IFK$(X)=""ANDC$<>"C"THEN150:ELSEIFLEN(K$(X)
)<>1THEN110:ELSELINEINPUT"NAME: ";N$(X):INPUT"PHONE NUMBER";P$(X
)
120 IFLEN(N$(X))=10THEN125:ELSEIFLEN(N$(X))>10THENPRINT"NAME TOO
 LONG ... REDO":GOTO110:ELSEFORY=1TO10-LEN(N$(X)):N$(X)=N$(X)+"
":NEXTY
125 N$(X)=K$(X)+"-"+N$(X):P$(X)=K$(X)+P$(X):LP=LEN(P$(X)):P$(X)=
P$(X)+STRING$(11-LP,0)
127 P$(X)=P$(X)+CHR$(&HBF):IFC$="C"THENGOSUB300:GOTO20
130 NEXTX
150 FORY=STOX-1:IFK$(S)=""THENS=0:GOTO20
155 N=Y/5
160 FORZ=1TO12:C=ASC(MID$(N$(Y),Z,1))
170 POKEP,C:P=P+1:NEXTZ:IFN<>INT(N)THENPOKEP,32:P=P+1
175 NEXTY:POKE&H84FF,X-1:POKEP,128
200 P=&H8200
210 IFPEEK(P)=255THEN220:ELSEP=P+1:GOTO210
220 FORY=STOX-1
230 FORZ=1TO12:C=ASC(MID$(P$(Y),Z,1))
240 POKEP,C:P=P+1:NEXTZ:NEXTY:POKEP,255
250 PRINT"NEW ENTRIES COMPLETE":PRINT"SAVE THE NEW SMARTCAT/EMTE
RM WITH TAPEDISK":END
300 P=&H8540:N=0:IFK$(X)=""THENRETURN
310 IFPEEK(P)=128THEN400:ELSEIFPEEK(P)<>ASC(K$(X))THENN=N+1:IFN<
>5THENP=P+13:GOTO310:ELSEN=0:P=P+12:GOTO310
320 FORZ=1TO12:C=ASC(MID$(N$(X),Z,1))
330 POKEP,C:P=P+1:NEXTZ
340 P=&H8200
350 IFPEEK(P)<>ASC(K$(X))THENP=P+12:GOTO350
360 FORZ=1TO12:C=ASC(MID$(P$(X),Z,1))
370 POKEP,C:P=P+1:NEXTZ:RETURN
400 PRINT"KEY ";K$(X);" NOT IN OIRECTORY":RETURN
500 P1=&H8540:P2=&H8200:N=0
505 CLS:PRINT"TELEPHONE DIRECTORY      PAGE";N/20+1:PRINT:PRINT:F
ORW=1TO10
510 FORY=1TO2:N=N+1
520 FORZ=1TO12
530 PRINTCHR$(PEEK(P1));:P1=P1+1:NEXTZ
540 IFN/5<>INT(N/5)THENP1=P1+1
550 PRINT" ";:P2=P2+1
560 FORZ=1TO10:PRINTCHR$(PEEK(P2));:P2=P2+1:NEXTZ:P2=P2+1:IFY=1T
HENPRINTTAB(32)" ";
570 IFN=55THENPRINT" ":GOTO580:ELSENEXTY:PRINT" ":NEXTW
580 G$="":INPUT"<ENTER> FOR OTHERS ... OR ENTER 'G ' TO GO ON";G
$:IFG$="G"THENPRINT@832,STRING$(60,32);:PRINT@831," ";:RETURN:EL
SEIFG$<>""THEN580
590 IFN<41THEN505:ELSE500
```

Program Listing 3. PHONEDIR/CAT

See List of Advertisers on Page 563

Long, Long Division

by David Cecil

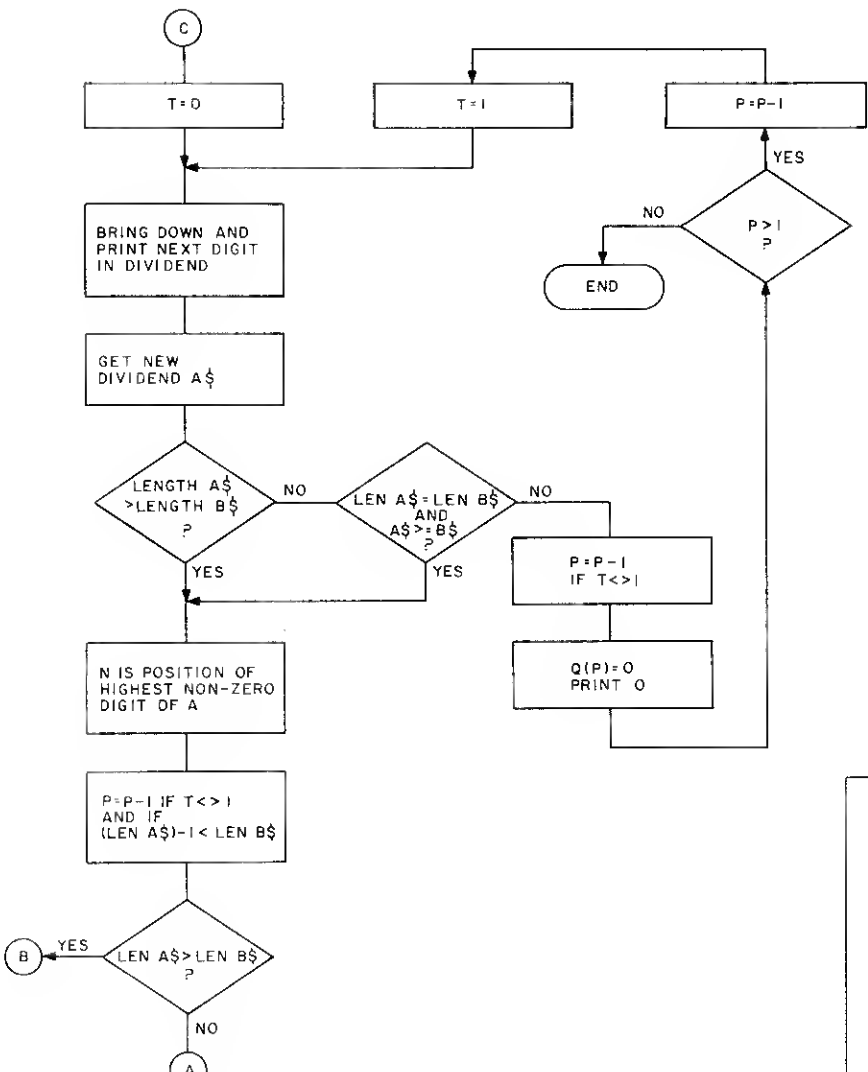


Figure 2

Teach your kids (or yourself) long division. Your Model I makes a good, patient tutor.

Remember the first elementary school teacher to torture you with long, long division problems? With this program you can have a total of 254 digits in the divisor and dividend, but for easy readability of the intermediate steps, restrict yourself to 64 digits.

The division is displayed on the screen just as on paper. The quotient appears at the top and the remainder on the last line. I chose not to draw lines after each two rows of figures. Instead, rows that would be under lines on paper begin with a zero. Figure 1 is

```
        1097
667 /732343
     667
     06534
      6003
      05313
       4669
       0644
```

Figure 1

The Key Box

Model I
16K RAM
Basic Level II

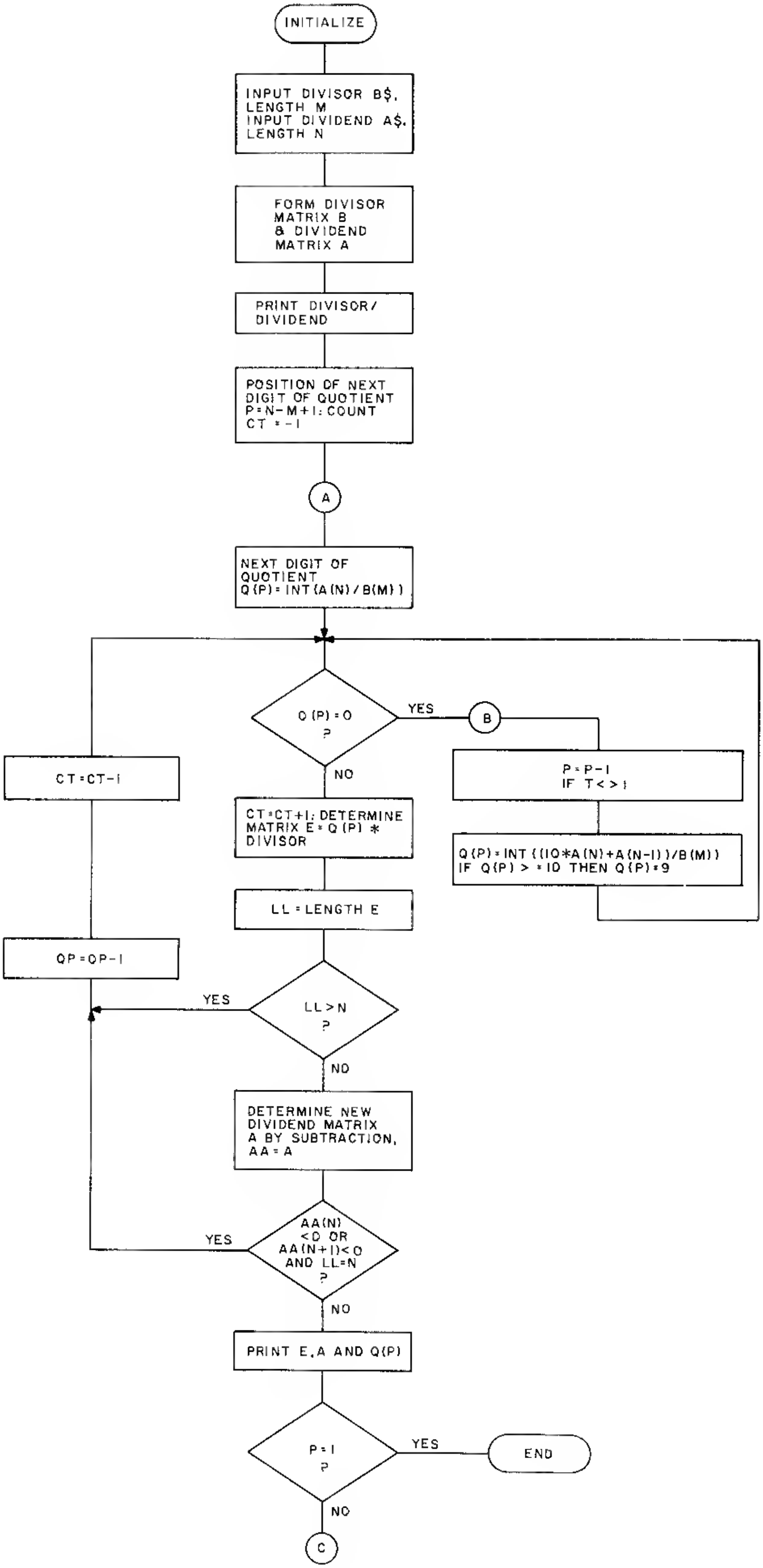


Figure 3

an example of long division.

PRINT@ statements are used to add digits to the quotient during division without scrolling the screen. If the division uses more than 14 lines of intermediate steps, the screen will scroll. We cannot scroll backward to see the quotient so the entire quotient is printed at the end whenever scrolling occurs.

The Program

After typing Run, the screen clears and prompts "Input divisor as a string, highest digit to lowest digit." After entering the divisor, a second prompt asks for the dividend.

If you enter a smaller dividend than divisor you get the error message "Illegal function call." To divide 17 into 13 use 17 for the divisor and 1300000000000 for the dividend. A word of caution: This program is not set up to handle decimals. The program will interpret a decimal as a zero; to the program 31.4 appears as 3104.

The line numbers in the Program Listing are in increments of 10. All the remark line numbers end in five; skip those lines when typing in the program. The lines have been printed for clarity but you need not indent when entering. The remark statements are keyed to the flowchart (Figs. 2 and 3) for your ease in following the program.

The divisor and dividend are entered as strings (lines 20 and 30). For arithmetic manipulations they are changed to single indexed matrices in lines 50 and 60. During solving, the dividend appears as the string A$ (lines 330–370 and 450–480) and at other times as the matrix A (lines 100–150 and 230–280, 340).

Subtraction is performed in lines 230–280. The dividend A is put in the temporary matrix AA and line 250 does the subtraction. Borrowing is handled by line 260. If the trial quotient digit Q(P) is too large, it is reduced by one in line 270 and the multiplication-subtraction sequence (lines 110–280) is redone.

Lines 590–640 and the variable DD are only needed for scrolling.

Since most intermediate dividends begin with one or more zeros, line 460 obtains the correct length of the dividend as a string with leftmost entry nonzero.

May long division no longer haunt you in your dreams. ■

David Cecil can be reached at the Dept. of Mathematics, Texas A & I University, Campus Box 172, Kingsville, TX 78363.

```
5 'INITIALIZE
10      CLEAR 5000:CLS:DEFINT A-Z
20        INPUT"INPUT DIVISOR AS A STRING,HIGBEST DIGIT TO LOWES
T DIGIT";B$
30        INPUT"INPUT DIVIDEND AS A STRING,HIGHEST DIGIT TO LOWE
ST DIGIT";A$
40      NZ=LEN(A$):N=NZ:M=LEN(B$):DIM A(N+1),AA(N+1),E(N+1),B(M)
,Q(N-M+1)
45 'PUT THE DIVISOR INTO MATHIX B
50      FOR I=1 TO M:B(M-I+1)=VAL(MID$(B$,I,1)):NEXT I
55 'PUT THE DIVIDEND INTO MATHIX A
60      FOR I=1 TO N:A(N-I+1)=VAL(MID$(A$,I,1)):NEXT I
70 CLS
80 PRINT@65,B$+" / "+A$;
85 'P IS THE POSITION OF THE NEXT DIGIT OF THE QUOTIENT
90      P=N-M+1:CT=-1
95 'Q(P) IS THE NEXT DIGIT OF THE QUOTIENT
100      Q(P)=A(N)/B(M)
110      IF Q(P)<>0 THEN 160
120        IF T<>1 THEN P=P-1
130      Q(P)=(10*A(N)+A(N-1))/B(M)
140        IF Q(P)>=10 THEN Q(P)=9
150      GOTO 110
155 'OBTAIN THE PRODUCT OF QUOTIENT Q(P) & DIVISOR B, PUT RESULT
 IN MATRIX E
160      CT=CT+1:C=0:FOR I=P TO P+M-1
170        E(I)=Q(P)*B(I-P+1)+C
180        C=E(I)/10
190        E(I)=E(I)-10*C
200      NEXT I
205 'LL IS THE LENGTH OF E
210      IF C<>0 THEN E(P+M)=C:LL=P+M:ELSE LL=P+M-1
220      IF LL>N THEN Q(P)=Q(P)-1:CT=CT-1:GOTO 110
225 'GET THE NEW DIVIDEND A BY SUBTRACTION
230      FOR J=P TO N:AA(J)=A(J):NEXT J
240      AA(N+1)=0
250      FOR J=P TO LL:AA(J)=AA(J)-E(J):K=J
260        IF K<=LL AND AA(K)<0 THEN AA(K)=AA(K)+10:AA(K+1)=AA(K
+1)-1:K=K+1:                GOTO 260
270        IF AA(N)<0 OR (AA(N+1)<0 AND LL=N) THEN Q(P)=Q(P)-1:C
T=CT-1:GOTO 110
280      NEXT J
285 'PRINT THE MATRIX E
290      A1=132+M+NZ-I+128*CT:IF A1>1023 THEN PRINT
300      FOR I=LL TO P STEP -1
310        A1=132+M+NZ-I+128*CT:GOSUB 590
320      PRINT@A1,RIGHT$(STR$(E(I)),1);:NEXT I
325 'PRINT THE NEW DIVIDEND A
330      A$=""
340      FOR K=N TO P STEP -1:A(K)=AA(K):A$=A$+RIGHT$(STR$(A(K))
,1):NEXT K
350      A2=A1+64-LEN(A$)+1
360        GOSUB 610
370      PRINT@A2,A$;
375 'PRINT THE NEXT DIGIT OF THE QUOTIENT
380      A3=4+M+NZ-P
390        IF DD=1 THEN 410
400      PRINT@A3,RIGBT$(STR$(Q(P)),1)·
405 'BAS UNIT'S PLACE DIGIT OF QUOTIENT BEEN PRINTED?
410      IF P=1 THEN  550  ELSE T=0
415 'PRINT TBE NEXT DIGIT IN THE DIVIDEND
420      NN=0:W=P-1
430        A4=A2+LEN(A$)+NN:GOSUB 630
440        PRINT@A4,RIGHT$(STR$(A(W)),1);
445 'CONPARE SIZE OF NEW DIVIDEND WITH DIVISOR
450      A$=A$+RIGHT$(STR$(A(W)),1)
460        Z$=LEFT$(A$,1):IF LEN(A$)=1 THEN 470  ELSE IF Z$="0"
THEN                          A$=MID$(A$,2,LEN(A$)-1):NN=NN+1:GO
TO 460
470      IF LEN(A$)>LEN(B$) THEN 530
480        IF LEN(A$)=LEN(B$) AND A$>=B$ THEN 530
485 'LOOP IF A LESS THAN B
490        IF T<>1 THEN P=P-1
500      Q(P)=0:IF DD=1 THEN 510  ELSE PRINT@4+M+NZ-P,"0";
510        IF P>1 THEN P=P-1 ELSE 550
520      T=1:W=P:GOTO 430
525 'GET POSITION OF HIGHEST NON-ZERO DIGIT OF A
530      N=N-NN:IF T<>1 AND (LEN(A$)-1)<LEN(B$) THEN P=P-1
540        IF LEN(A$)>LEN(B$) THEN 120  ELSE 100
545 'FINAL PRINTING
550      IF DD=1 THEN 560  ELSE PRINT@890,""
560        IF DD<>1 THEN END
570      PRINT:PRINT"QUOTIENT= ";:FOR J=NZ-M+1 TO 1 STEP -1:PRIN
T                         RIGHT$(STR$(Q(J)),1);:NEXT J
580      END
585 'THE FOLLOWING LINES SCROLL THE SCREEN
590        IF A1>1023 THEN DD=1:A1=A1-64:GOTO 590
600      RETURN
610        IF A2>1023 THEN DD=1:A2=A2-64:GOTO 610
620      IF DD=1 THEN PRINT:RETURN ELSE RETURN
630        IF A4>1023 THEN DD=1:A4=A4-64:GOTO 630
640      RETURN
```

Program Listing 1

✓See List of Advertisers on Page 563

Almazar I

by Winston Llamas

As you visit the 72 rooms of Almazar I, you'll find treasures and run into danger, and if you're lucky, you'll reach the next dimension.

Almazar Part I is the first part of a multi-part adventure known simply as Alamazar. In Part I, the adventurer's goal is to be transported into the next dimension. The game might lack the refinement of programs such as Zork, but it is more advanced than typical adventure games written in Basic. Almazar starts similarly to the original Adventure program, and it provides ample puzzles for the adventurer to solve.

The game has seventy-two rooms; some rooms are filled with treasures, while others are replete with danger. A familiarity with various books and movies will be helpful in achieving success, yet there are numerous occasions in which common sense alone can lead the adventurer out of trouble.

Almazar is divided into seven separate programs. One is the Almazar main program, which is loaded when the user wishes to play the game. The other programs create the data base necessary for the program to run. This version of Almazar was designed for a one-drive 32K Model III. The modifications needed to make the program run on a one-drive Model I are outlined below.

Modifications

One-drive Model I owners must use two disks to hold the program (unless the drives are either 80 tracks or the system is running double density, or both, allowing everything to fit in one disk). You must also add the line:

```
60 INPUT"REPLACE DISK AND
HIT ENTER WHEN READY";G$
```

The first disk should contain the Almazar main program as well as the SFZVOC, SFAMOVE, SFAOBDES data bases while the second disk should contain SFALDES, SFASDES, and

The Key Box

Model I or III
32K RAM
Disk Basic
1 Disk Drive

MO	possible moves
VC$	command file verbs
TC$	command file objects
VC	command verb identifier
TC	command object identifier
OP	designates location of object
LO$	long object descriptions
SO$	short object descriptions
LC	length of command line
V1	length of first word
C1	location of space in command
OW	indicates one word command
IV	index to verb
IO	index to object
S2	initial positions of objects
PR	points subtracted for hints
LI	turns remaining before lamp goes out
TU	number of turns remaining in the game
FT	indicates if room has been visited
SR	indicates special rooms
KG	indicates if gnome is dead
SC	flag gnome chase
W1	flag dimming lamp
W2	flag limited turns
EN	indicates if there is something to enter
LE	indicates possibility of leaving
DI	indicates how many times player has died
IN	number of items carried
PT	points adventurer scored
NF$	name of file for long or short room descriptions
SP	special situations array

Table 1. Variable Listings and Explanations

> *"The game has 72 rooms; some rooms are filled with treasures, while others are replete with danger"*

Program Listing 1. The Search for Almazar, Part 1—The Proving Ground

```
10 CLS:PRINTTAB(18)"The Search for Almazar: Part 1":PRINTTAB(24)
"The Proving Ground":PRINT
20 PRINT"   This program is the first of a projected series of p
rograms
whose central theme is the continuing search for the super being
Almazar.  This game, however, will play to a satisfactory ending
if the player does not wish to continue";
30 PRINT" the series.":PRINT:PRINT"   The game begins at one end
 of a road, by an old abandoned    shack.  (Just like the origin
```

Listing 1 continues

Command	**Action**
N	adventurer moves northward
Light Lamp	lamp is lit
Off Lamp	lamp is turned off
Enter	adventurer enters something
Inventory	prints objects adventurer is carrying
Hint or Help	asks for hints
Look	ask for new room description
Throw Knife	throws knife
Take object	adventurer takes object
Drop object	adventurer drops object

Table 2

SFARESP. The Model III uses the uppercase @ as a closing single quote, and this must be modified for Model I use.

Almazar's design lets you change it easily. If you want to expand the vocabulary, the changes in SFAVOC and in the main program are simple. Likewise, all the other data bases can be altered by changing the data-base-generating programs. A word of caution: Since the data bases SFALDES, SFASDES, and SFARESP are random-access files, you must be careful to limit the text to 256 characters.

Running Almazar

The game isn't easy, since you must gather various treasures before you move to the next dimension. In addition, numerous traps and puzzles catch the unwary. ■

Winston Llamas is a student at Rensselaer Polytechnic Institute where he's majoring in computer science. His address is 121 North Gordon St., Alexandria, VA 22304.

Listing 1 continued

```
al ADVENTURE program.)  The player will then explore the many di
fferent scenarios in the game.  Butbe careful.  There are many";

40 PRINT" traps to catch the careless and     unthinking adventu
rer."
45 I$=INKEY$:IFI$<>""THEN50ELSE45
50 CLS:PRINT"The program accepts one or two word commands.  Some
 examples:
   To take an object, type TAKE OBJECT or T OBJECT for short.
   To go north, type NORTH or N for short.
   To see what you're carrying, type INVE(NTORY)."
60 PRINT"   Type SCORE and the program will give you your curren
t score.   Type SAVE and the program will save the game for lat
er play.
   Type QUIT and the game will be terminated.
   To light or turn off a lamp, type LIGHT LAMP or OFF LAMP."
70 PRINT"   To get a description of the room, type LOOK.":PRINT:
PRINT"Other commands include SMASH, TOSS, SHOW, CROSS, etc.":PRI
NT"In addition, one may type ENTER to enter a shack or type LEAV
E
to leave a snack.  Of course, if there is no";
80 PRINT" way out you cannot leave.  A hint - caves are dark and
 often dangerous."
85 I$=INKEY$:IFI$<>""THEN90ELSE85
90 CLS:PRINT"   You have a limited number of turns in which to a
ccomplish    your task.  If you don't finish on time, you will..
.  From time to time you may need a hint, so just type HINT or H
ELP.  But    remember, nothing is free in ";
100 PRINT"the world today.":PRINT:PRINT"   Good luck and may Alm
azar guide you to a safe journey."
105 I$=INKEY$:IFI$=""THEN105ELSECLS:RUN"SFAMAIN:0"
```

Program Listing 2. Main Program

```
10 CLS:PRINT@320,CHR$(23);"      The Search for Almazar":PRINTTAB
(13)"Part 1":PRINT:PRINT"  Written by Winston M. Llamas":PRINT:P
RINT"c. Aug 1981  All Rights Reserved"
20 CLEAR2000:DIMFT(72),MO(72,5),VC$(53),VC(53),TC$(42),TC(42),OP
(29),LO$(29),SO$(29),SP(14)
30 OPEN"I",1,"SFAMOVE:0":FORX=1TO72:INPUT#1,RI,MO(X,0),MO(X,1),M
O(X,2),MO(X,3),MO(X,4),MO(X,5):NEXT:CLOSE1
40 OPEN"I",1,"SFAVOC:0":FORX=0TO53:INPUT#1,VC$(X),VC(X):NEXTX:FO
RY=0TO42:INPUT#1,TC$(Y),TC(Y):NEXTY:CLOSE1
50 OPEN"I",1,"SFAOBDES:0":FORX=0TO29:INPUT#1,DU,LO$(X):NEXTX:FOR
Y=0TO19:INPUT#1,DU,SO$(Y):NEXTY:FORZ=0TO29:INPUT#1,OP(Z):NEXTZ:C
LOSE1
70 CLS:N=1:GOSUB7000
100 IFRN>10ANDRN<30GOSUB7400
114 TU=TU+1:IFTU>400THENRE=74:PRINT:GOSUB7100:DI=3:GOTO7300
115 IFSP(8)=1THENSP(14)=SP(14)+1:ELSEIFSP(8)=0THENSP(14)=0
116 IFSP(2)=1THENSP(1)=SP(1)-1:IFSP(1)<1THENSP(2)=0
117 IFSP(1)<20ANDSP(2)=1ANDOP(8)=-1ANDW1<>1THENRE=17:PRINT:GOSUB
7100:W1=1
118 IFSP(13)>0THENSP(13)=SP(13)+1:IFSP(13)>5THENRE=70:PRINT:GOSU
B7100:GOSUB7200:GOTO100
119 IFTU>360ANDW2<>1THENRE=16:PRINT:GOSUB7100:W2=1
121 IFRN=15THENSC=1:ELSEIFOP(23)=RNTHENSP(6)=SP(6)+1:IFSP(6)>5TH
ENIO=22:PRINT:GOSUB1800
122 IFSP(14)>7THENRE=65:PRINT:GOSUB7100:SP(14)=0:SP(8)=0:GOSUB72
00:GOTO100
123 C1=0:OW=0:PRINT:INPUTCO$:PRINT:LC=LEN(CO$):FORX=1TOLC:IFMID$
(CO$,X,1)=" "THENC1=X:GOTO125:ELSENEXT
124 IFCO$=""THENRE=2:GOSUB7100:GOTO100
125 V1=C1-1:IFC1=0ORV1>4THENV1=4:ELSEOW=1
130 C1$=LEFT$(CO$,V1):FORX=0TO53:IFC1$=VC$(X)THENIV=VC(X):GOTO14
0:ELSENEXT:RE=2:GOSUB7100:GOTO100
140 IFIV>16THEN970:ELSEIFIV>6THEN190
144 IFIV=1ANDSP(7)<>2ANDRN=62THENRE=91:GOSUB7100:GOTO100
145 IFIV=1ANDRN=59ANDOP(29)<>59THENRE=1:GOSUB7100:GOTO100
150 N=MO(RN,(IV-1)):IFN=0THENRE=1:GOSUB7100:GOTO100:ELSEGOSUB700
0:GOTO100
190 ONIV-6GOSUB200,300,400,500,600,700,800,900,3300,3400:GOTO100

200 N=RN:FT(N)=0:GOSUB7000:RETURN
300 IFRN=47ORRN=43ORRN=52ORRN=63ORRN=54ORRN=61ORRN=70ORRN=69THEN
305:ELSERE=102:GOTO390
305 INPUT"THE HINT WILL COST YOU 3 POINTS. DO YOU STILL WANT IT"
;HQ$:IFLEFT$(HQ$,1)="N"THENRETURNELSEPRINT
310 IFRN=47THENRE=94
315 IFRN=63THENRE=95
320 IFRN=54THENRE=96
325 IFRN=61THENRE=97
330 IFRN=70THENRE=98
335 IFRN=69THENRE=99
340 IFRN=52THENRE=100
345 IFRN=43THENRE=101
385 PR=PR+3
390 GOSUB7100:RETURN
400 CA=0:PRINT"YOU ARE CARRYING:":FORX=0TO29:IFOP(X)=-1THENPRINT
SO$(X):CA=1
410 NEXTX:IFCA=0THENPRINT"NOTHING"
420 RETURN
500 RE=6:IFRN=27ORRN=28THENRE=43
505 IFRN=41THENRE=48
510 IFRN=43ORRN=44ORRN=49THENRE=30
520 GOSUB7100:IFRE=30THENRE=31:GOSUB7100
530 IFRE<>6THEN7200ELSERETURN
600 PT=0:FORX=0TO7:IFOP(X)=-1THENPT=PT+7
610 IFOP(X)=72THENPT=PT+10
620 NEXTX:PT=PT-(DI*10)-PR:IFRE=8THENPT=PT+10
630 PRINT"YOU HAVE SCORED ";PT;"POINTS.":RETURN
700 INPUT"ARE YOU SURE YOU WANT TO QUIT";QU$:IFLEFT$(QU$,1)<>"N"
THEN7300ELSERETURN
800 OPEN"O",1,"SFA/SAV:0":FORX=0TO29:PRINT#1,OP(X)",";:NEXTX:FOR
Y=1TO72:PRINT#1,FT(Y)",";:NEXTY:FORZ=0TO14:PRINT#1,SP(Z)",";:NEX
TZ:PRINT#1,RN","TU","SC","IN","PR:CLOSE1
810 PRINT"THE GAME IS NOW SAVED FOR LATER USE. TO CONTINUE THE S
AME GAME,
TYPE `RESTORE' ON YOUR FIRST TURN AFTER YOU RESUME PLAY.":PRINT:
GOSUB600:PRINT:END
900 IFTU>1THENPRINT"IT'S TOO LATE TO RESTORE AN OLD GAME.":RETUR
N
910 OPEN"I",1,"SFA/SAV:0":FORX=0TO29:INPUT#1,OP(X):NEXTX:FORY=1T
O72:INPUT#1,FT(Y):NEXTY:FORZ=0TO14:INPUT#1,SP(Z):NEXTZ:INPUT#1,R
```

Listing 2 continues

```
N,TU,SC,IN,PR:CLOSE1:N=RN:GOSUB7000:RETURN
970 S2=C1+1:C2$=MID$(CO$,S2,4)
980 FORX=0TO42:IFC2$=TC$(X)THENIO=TC(X):GOTO990:ELSENEXTX:RE=2:G
OSUB7100:GOTO100
990 ONIV-16GOSUB1000,1100,1200,1200,1400,1500,1600,1700,1800,190
0,2000,2100,2200,2300,2400,2500,2600,2700,2800,2900,3000,3100,32
00:GOTO100
1000 IFIO>18THENRE=2:GOTO1030
1005 IFIN+1>7THENRE=83:GOTO1099
1010 IFIO=18THENIFOP(5)=RNTHENRE=3:OP(5)=-1:GOTO1098:ELSEIFOP(19
)=RNTHENRE=3:OP(19)=-1:GOTO1098
1015 IFIO<>5ANDIO<>16THEN1018:ELSEIFIO=5THENIFOP(5)=-1THENRE=7:G
OTO1099:ELSEIFOP(19)=-1THENRE=62:GOTO1099
1016 IFIO=5THENIFOP(22)=RNTHENIFOP(18)=-1THENRE=3:OP(22)=0:OP(18
)=0:OP(5)=-1:GOTO1099:ELSERE=60:GOTO1099
1017 IFIO=16THENIFOP(16)=RNTHENIFSP(3)=1THENRE=3:OP(16)=-1:GOTO1
098:ELSERE=18:GOTO1099
1018 IFIO=1ANDOP(20)=RNTHENRE=23:GOSUB7100:GOTO7200
1020 IFOP(IO)=RNTHENRE=3:OP(IO)=-1:ELSEIFOP(IO)=-1THENRE=7ELSERE
=4
1030 IFIO=28THENIFOP(8)<>-1THENRE=63:ELSEIFRN<>8THENRE=71:ELSERE
=3:SP(1)=100
1040 IFIO=29THENIFOP(5)=-1OROP(19)=-1THENRE=62:ELSEIFOP(18)<>-1T
HENRE=60:ELSEIFRN<>43ANDRN<>44ANDRN<>49THENRE=4:ELSERE=3:OP(18)=
0:OP(19)=-1
1045 IFIV=22THENIFRE=60THENRE=77:ELSEIFRE=4THENRE=78
1046 IFIV=22ANDRE=71THENRE=78
1050 IFIO=7ANDOP(26)=RNTHENOP(7)=-1:OP(26)=0:RE=3
1060 IFIO=27THENIFRN=32THENRE=9ELSERE=4
1070 IFIO=25THENIFRN=65THENRE=9ELSERE=4
1080 IFIO=2ANDOP(23)=RNTHENRE=93
1085 IFIO=31THENIFRN=69THENRE=9ELSERE=4
1086 IFIO=19THENIFRN=52THENRE=9ELSERE=4
1090 IFIO=3ANDOP(24)=RNTHENRE=61
1098 IFRE=3ANDIO<19THENIN=IN+1
1099 GOSUB7100:RETURN
1100 IFIO<19ORIO=29THEN1105:ELSERE=2:GOTO1199
1105 IFOP(IO)=-1ANDIO<>7THENOP(IO)=RN:RE=3:ELSERE=5
1106 IFIO=16ANDRE=3THENRE=66:SP(3)=0
1110 IFIO=18ANDOP(5)=-1THENOP(5)=RN:RE=3:GOTO1140
1120 IF(IO=18ORIO=29)ANDOP(19)=-1THENOP(19)=RN:RE=3
1130 IFIO=7ANDOP(IO)=-1THENIFOP(14)=RNTHENRE=3:OP(7)=RN:ELSERE=7
5:OP(7)=0:IN=IN-1
1140 IFRE=3THENIN=IN-1
1199 GOSUB7100:RETURN
1200 IFIO>18THENRE=2:GOTO1299
1210 IFOP(IO)=-1THENRE=6:ELSERE=5:GOTO1299
1220 IFIV=19AND(RN=43ORRN=44)ANDIO=12THENSP(8)=ABS(SP(8)-1):IFSP
(8)=1THENRE=28ELSERE=29
1230 IFIV=19AND(RN=27ORRN=28)ANDIO=11THENSP(9)=ABS(SP(9)-1):IFSP
(9)=1THENRE=35ELSERE=36
1240 IFIV=20ANDIO=10THENGOSUB1300
1299 GOSUB7100:RETURN
1300 IFRN=61ANDSP(5)=0THENSP(5)=1:RE=22:GOSUB7100:RE=25:PRINT:OP
(2)=RN:OP(23)=0:RETURN
1400 RE=4:IFRN=25ANDIO=26THENRE=34
1410 IFRN=32ANDIO=26THENRE=37
1420 IFRN=43ANDIO=35THENRE=32
1430 IFRN=44ANDIO=26THENRE=67
1440 IFRN=65ANDIO=25THENIFSP(10)=1THENRE=38ELSERE=44
1450 IFRN=70ANDIO=26THENRE=33
1454 IFIO<>26ANDIO<>25ANDIO<>35THENRE=2
1455 IFRN<>70ORIO<>26THEN1460
1456 FORX=0TO7:IFOP(X)<>72THEN1460
1457 NEXTX:IFOP(10)=-1THENRE=8:GOSUB7100:GOTO7300
1460 GOSUB7100:IFRE=44THEN7200
1470 RETURN
1500 IFIO=8THENIO=28:GOTO1000
1510 IFIO=18THENIO=29:GOTO1000
1520 RE=2:GOSUB7100:RETURN
1600 RE=2:IFIO<>8ANDIO<>13THEN1690
1605 IFSP(2)=1ANDIO=8THENRE=82:GOTO1690
1610 IFOP(IO)<>-1THENRE=5:GOTO1690
1620 IFIO=8ANDSP(1)<1THENRE=13:GOTO1690
1630 IFOP(17)<>-1THENRE=12:GOTO1690
1640 IFIO=8THENRE=3:SP(2)=1:SP(13)=0:ELSEOP(13)=0:IN=IN-1:IFRN=5
2THENOP(20)=0:OP(1)=RN:RE=24:ELSERE=3
1690 GOSUB7100:RETURN
1700 RE=6:IFIO=20ORIO=22ORIO=36ORIO=27THEN1705:ELSERE=2:GOTO1690

1705 IFIO=36THENIFOP(21)<>RNTHENRE=4
1706 IFIO=20ANDOP(24)<>RNTHENRE=4
1707 IFIO=22ANDOP(23)<>RNTHENRE=4
1708 IFRE=4THEN1690
1710 INPUT"WITH WHAT, YOUR BARE HANDS";QU$:IFLEFT$(QU$,1)="N"THE
N1750
1720 IFIO=20THENRE=21
1730 IFIO=22THENRE=45
1740 IFIO=36THENRE=81
1745 IFIO=27THENRE=11
1750 PRINT:GOSUB7100:IFRE<>21ANDRE<>45THENRETURNELSE7200
1800 RE=2:IFIO=27THENIFOP(25)=RNTHENRE=10ELSERE=4
1810 IFIO=22ANDOP(23)=RNTHENSP(6)=0:RE=45:GOSUB7100:GOTO7200
1811 IFIO>19THEN1820
1815 IFOP(IO)<>-1THENRE=5:ELSEIFOP(IO)<>RNTHENRE=4
1816 IFOP(IO)=-1THENOP(IO)=RN:RE=3:IN=IN-1
1820 GOSUB7100:RETURN
1900 RE=6:IFIO>18ANDIO<>27THENRE=2:GOTO1960
1910 IFIO=27THENIFOP(25)=RNTHEN1710ELSERE=4:GOTO1960
1920 IFIO=7ANDOP(7)=-1THENRE=75:OP(7)=0
1930 IF(IO=18ORIO=5)ANDOP(5)=-1THENRE=84:GOSUB7100:PRINT:RE=73:O
P(5)=0:IN=IN-1:GOTO1960
1940 IFIO=18ANDOP(19)=-1THENOP(19)=0:IN=IN-1:RE=84
1950 IFOP(IO)<>-1THENRE=5
1955 IFIO=18ANDOP(18)=-1THENIN=IN-1:OP(18)=0:RE=84
1960 GOSUB7100:RETURN
2000 RE=2:IFIO>18THEN2099
2010 IFIO=18ORIO=5THEN1900
2020 IFOP(IO)<>-1THENRE=5:GOTO2099:ELSERE=3:IN=IN-1:OP(IO)=RN:IF
(IO=6ANDOP(21)=RN)THEN2030:ELSE2099
2030 KG=RND(0)-(IN*.01):IFKG>.4THENPRINT"YOU KILLED A NASTY, KNI
FE THROWING GNOME!":OP(21)=0:ELSEPRINT"YOU MISSED! YOU OUGHT TO
GET YOU EYES EXAMINED."
2040 RETURN
2099 GOSUB7100:RETURN
2100 RE=19:IFIO<>5THEN2110:ELSEIFOP(5)=-1OROP(5)=RNTHENRE=40:OP(
5)=0:OP(18)=-1:GOTO2199
2101 IFOP(5)=RNTHENRE=40:OP(5)=0:OP(18)=RN
2105 IFOP(22)=RNTHENOP(22)=0:RE=40:GOTO2199
```

Listing 2 continued

```
2110 IFIO=15THENIFOP(15)=-1THENOP(15)=0:IN=IN-1:RE=80:ELSEIFOP(1
5)=RNTHENOP(15)=0:RE=80
2120 IFIO=16THENIFOP(16)=-1THENRE=41:IN=IN-1:OP(16)=0:GOSUB7100:
GOTO7200:ELSEIFOP(16)=RNTHENRE=66
2199 GOSUB7100:RETURN
2200 RE=57:IFIO<>29THENRE=2:GOTO2199
2210 IFOP(19)=-1THENRE=56:OP(19)=0:OP(18)=-1
2215 IFOP(19)=RNTHENRE=56:OP(19)=0:OP(18)=RN
2220 IFRN=43ORRN=44ORRN=49THENRE=56
2230 GOTO2199
2300 IFIO<>16ANDIO<>20THENRE=2:GOTO2199
2310 IFIO=16THENIFOP(16)<>RNTHENRE=4:GOTO2199:ELSEIFOP(15)=-1THE
NSP(3)=1:RE=3:GOTO2199:ELSERE=19:GOTO2199
2320 IFOP(24)<>RNTHENRE=4:GOTO2199:ELSEIFOP(16)<>-1THENRE=19:GOT
O2199
2330 IN=IN-1:OP(24)=0:RE=20:GOSUB7100:OP(16)=0:RE=26:PRINT:OP(3)
=RN:GOTO2199
2400 RE=6:IFIO>18ANDIO<>24THENRE=2:GOTO2199
2410 IFIO<>24THEN2199:ELSEIFRN<>59THENRE=4:GOTO2199
2415 IFOP(29)=59THENRE=85:OP(29)=0:GOTO2199
2420 RE=72:OP(29)=59:GOTO2199
2500 IFIO<>30ANDIO<>31ANDIO<>32THENRE=2:GOTO2199
2501 IFIO=30ANDRN=62THEN2530
2505 RE=4:IFIO=32ANDRN=66THENIFOP(9)=-1THENSP(11)=1:RE=3:ELSERE=
14
2510 IFIO=31ANDRN=69THENIFOP(9)=-1THENRE=3:SP(12)=1:ELSERE=14
2520 GOTO2199
2530 INPUT"LEFT";N1$:INPUT"RIGHT";N2$:INPUT"LEFT";N3$:PRINT
2540 IFVAL(N1$)=3ANDVAL(N2$)=7ANDVAL(N3$)=21THENRE=87:SP(7)=1:EL
SERE=89
2550 GOTO2199
2600 RE=2:IFIO<>21ANDIO<>23THEN2199
2610 RE=4:IFIO=21THENIF(RN<>27ANDRN<>28)THEN2199:ELSEIFSP(9)=0TH
ENRE=42:GOTO2199
2620 IFIO=23THENIF(RN<>43ANDRN<>44)THEN2199:ELSEIFSP(8)=0THENRE=
27:GOTO2199
2630 IFRN=43THENN=44:ELSEIFRN=44THENN=43
2640 IFRN=27THENN=28:ELSEIFRN=28THENN=27
2650 GOTO7000
2700 RE=2:IFIO<>23ANDIO<>29THEN2720
2710 IFRN=43ORRN=44ORRN=49THENRE=31:ELSERE=57
2720 GOSUB7100:IFRE=31THEN7200ELSERETURN
2800 IFIO<>30ANDIO<>31ANDIO<>32ANDIO<>33THENRE=2:GOTO2899
2810 RE=4:IFIO<>30THEN2820:ELSEIFRN<>62THEN2899:ELSEIFSP(7)<1THE
NRE=88:GOTO2899:ELSEIFSP(7)=2THENRE=68:GOTO2899:ELSERE=3:SP(7)=2
:GOTO2899
2820 IFIO=31ANDRN=69THENIFSP(12)<1THENRE=86:ELSEIFSP(12)=2THENRE
=68:ELSERE=39:SP(12)=2
2830 IFIO=32ANDRN=66THENIFSP(11)<1THENRE=86:ELSEIFSP(11)=2THENRE
=68:ELSESP(11)=2:IFOP(4)=0THENOP(4)=RN:RE=76:ELSERE=3
2840 IFIO=33THENIFRN<>65THEN2899:ELSEIFSP(10)=1THENRE=3:SP(10)=0
:OP(27)=RN:OP(28)=0:ELSERE=68
2899 GOSUB7100:RETURN
2900 IFIO<>30ANDIO<>31ANDIO<>32ANDIO<>33THENRE=2:GOTO2899
2910 RE=4:IFIO=30ANDRN=62THENIFSP(7)=0THENRE=69:ELSERE=3:SP(7)=0

2920 IFIO=31ANDRN=69THENIFSP(12)=0THENRE=69:ELSERE=3:SP(12)=0
2930 IFIO=32ANDRN=66THENIFSP(11)=0THENRE=69:ELSERE=3:SP(11)=0
2940 IFIO=33ANDRN=65THENIFSP(10)=1THENRE=69:ELSERE=3:SP(10)=1:OP
(27)=0:OP(28)=RN
2950 GOTO2899
3000 IFIO<>8THENRE=2:GOTO2899
3010 IFOP(IO)<>-1THENRE=5ELSEIFSP(2)<>0THENSP(2)=0:RE=3:ELSERE=9
2
3020 GOTO2899
3100 RE=6:IFIO>18ANDIO<>34THENRE=2:GOTO3199
3110 IFIO<>34THEN3199
3120 IFRN<>60THENRE=4:GOTO3199:ELSERE=79:N=61
3199 GOSUB7100:IFRE<>79THENRETURN:ELSEPRINT:GOTO7000
3200 IFIO<>18THENRE=2:GOTO2899
3205 RE=5:IFOP(18)=-1THENRE=90
3210 IFOP(5)=-1THENOP(5)=0:OP(18)=-1:RE=73
3220 IFOP(19)=-1THENRE=3:OP(19)=0:OP(18)=-1
3230 GOTO2899
3300 EN=0:IFRN=1ORRN=10ORRN=55THENN=RN+1:EN=1
3305 IFRN=62THENIFSP(7)=2THENN=72:EN=1:ELSERE=91:GOTO2199
3310 IFRN=31THENN=33:EN=1
3320 IFRN=51THENN=70:EN=1
3330 IFEN=0THENRE=58:GOSUB7100:ELSEGOSUB7000
3340 RETURN
3400 LE=0:IFRN=2ORRN=11ORRN=56THENN=RN-1:LE=1
3410 IFRN=33THENN=31:LE=1
3420 IFRN=70THENN=51:LE=1
3425 IFRN=72THENN=62:LE=1
3430 IFLE=0THENRE=59:GOSUB7100:ELSEGOSUB7000
3440 RETURN
7000 IF((RN=43ANDN=44)OR(RN=44ANDN=43))ANDSP(8)=0THENRE=27:GOSUB
7100:N=RN:RETURN
7001 IF((RN=27ANDN=28)OR(RN=28ANDN=27))ANDSP(9)=0THENRE=42:GOSUB
7100:N=RN:RETURN
7002 IFRN=61ANDN<>61ANDIV=1ANDSP(5)=0THENRE=64:GOSUB7100:N=RN:RE
TURN
7003 IFRN=67ANDN=71ANDSP(5)=0THENRE=1:GOSUB7100:N=RN:RETURN
7004 SR=0:IF((N>11ANDN<>15ANDN<30)OR(N>32ANDN<39))THENSR=1
7005 IFSR=0THEN7009
7006 IFSP(2)=1AND(OP(8)=-1OROP(8)=N)THEN7009:ELSERE=15:GOSUB7100
:IFSP(13)=0THENSP(13)=1
7007 RN=N:N=0:RETURN
7009 IFFT(N)=0THENNF$="SFALDES:0"ELSENF$="SFASDES:0"
7010 OPEN"R",1,NF$:FIELD1,255ASDE$:GET1,N:CLOSE1:SP(13)=0
7015 FORX=1TO255:IFMID$(DE$,X,10)="          "THEN7040
7030 PRINTMID$(DE$,X,1);:NEXTX
7040 PRINT:RN=N:FT(N)=1:N=0:IFRN=41ANDVO=0THENRE=55:PRINT:GOSUB7
100:VO=1
7050 FORX=0TO29:IFOP(X)=RNANDX<>21THENPRINT:PRINTLO$(X)
7060 NEXTX:RETURN
7100 OPEN"R",1,"SFARESP:0":FIELD1,255ASRE$:GET1,RE:CLOSE1:FORX=1
TO255:IFMID$(RE$,X,12)="            "THEN7110:ELSEPRINTMID$(RE$,
X,1);:NEXTX
7110 PRINT:RETURN
7200 DI=DI+1:IFDI=3THENRE=49:PRINT:GOSUB7100:GOTO7300
7205 PRINT:INPUT"DO YOU WANT ANOTHER CHANCE";QU$:IFLEFT$(QU$,1)<
>"Y"THEN7300
7210 IN=0:PRINT:RE=DI+45:GOSUB7100:FORX=0TO18:IFOP(X)=-1THENOP(X
)=INT(RND(15))
7220 NEXTX:OP(8)=4:OP(17)=6:OP(12)=42:N=1:SP(2)=0:SP(8)=0:SP(14)
=0:SC=0:PRINT:GOSUB7000:RETURN
7300 PRINT:GOSUB600:RE=50:IFPT>7THENRE=51
```

Listing 2 continues

Listing 2 continued

```
7320 IFPT>35THENRE=52
7330 IFPT>79THENRE=53
7340 IFPT=90THENRE=54
7350 PRINT:GOSUB7100:PRINT:END
7400 IFSC=ØTHENRETURN
7410 IFOP(21)=ØTHENOP(21)=INT(RNO(35))
7420 IFOP(21)>RNTHENOP(21)=OP(21)-1
7425 IFOP(21)<RNTHENOP(21)=OP(21)+1
7430 IFOP(21)<>RNTHENRETURN
7440 PRINT:PRINTLO$(21):RA=RND(Ø)+(IN*.Ø1):IFRA>.7THENPRINT:PRIN
T"HE THROWS A SMALL KNIFE AT YOU!":PRINT:IFRA>.9THENPRINT"IT GET
S YOU!":GOTO72ØØ:ELSEPRINT"IT MISSES YOU!"
7450 RETURN
```

Program Listing 3

```
1Ø OPEN"O",1,"SFAMOVE:Ø"
2Ø FORX=1TO72:READN1,N2,N3,N4,N5,N6
3Ø PRINT#1,X","N1","N2","N3","N4","N5","N6:NEXTX:CLOSE1
4Ø DATA7,2,Ø,3,Ø,Ø,Ø,Ø,Ø,1,Ø,Ø,6,1,Ø,4,Ø,Ø,5,3,Ø,Ø,Ø,Ø,Ø,6,4,Ø,Ø
,Ø,Ø,Ø,3,5,Ø,Ø,9,8,1,Ø,Ø,Ø,32,Ø,Ø,7,Ø,Ø,3Ø,Ø,7,1Ø,Ø,Ø,Ø,9,Ø,11,Ø
,Ø,Ø,1Ø,12,29,Ø,Ø,11,Ø,13,Ø,Ø,Ø
5Ø DATA12,Ø,Ø,14,Ø,Ø,Ø,13,Ø,15,Ø,Ø,Ø,14,16,17,Ø,Ø,15,Ø,Ø,Ø,28,Ø,
Ø,15,Ø,18,Ø,Ø,18,18,19,23,18,18,18,19,19,2Ø,19,17,21,19,2Ø,22,2Ø
,2Ø,Ø,Ø,2Ø,Ø,Ø,Ø,23,2Ø,22,22,22,22,23,18,22,23,24,23,Ø,25,23,Ø,Ø
,Ø
6Ø OATAØ,26,Ø,24,Ø,Ø,27,29,Ø,25,Ø,Ø,28,Ø,26,Ø,Ø,Ø,Ø,Ø,27,Ø,Ø,16,
Ø,11,Ø,26,Ø,Ø,39,31,9,Ø,Ø,Ø,Ø,33,32,3Ø,Ø,Ø,31,Ø,8,Ø,Ø,Ø,38,36,34
,31,33,33,33,35,34,34,34,34,36,35,35,34,35,35,37,36,35,33,36,36
7Ø OATA37,37,36,38,37,37,38,37,33,38,38,38,4Ø,Ø,3Ø,Ø,Ø,Ø,43,41,3
9,Ø,Ø,Ø,42,Ø,4Ø,Ø,Ø,Ø,Ø,Ø,41,Ø,Ø,44,Ø,4Ø,Ø,Ø,Ø,45,Ø,43,Ø,Ø,Ø,5
Ø,46,44,48,Ø,Ø,Ø,47,Ø,45,Ø,Ø,54,Ø,Ø,46,Ø,Ø,51,45,49,Ø,Ø,Ø
8Ø DATA48,Ø,Ø,Ø,Ø,Ø,55,53,45,51,Ø,Ø,7Ø,5Ø,48,52,Ø,Ø,Ø,51,Ø,Ø,Ø,Ø
,Ø,54,Ø,5Ø,Ø,Ø,Ø,Ø,47,53,Ø,Ø,56,Ø,5Ø,Ø,Ø,Ø,63,59,55,57,Ø,Ø,58,56
,Ø,Ø,Ø,Ø,Ø,Ø,57,Ø,Ø,Ø,6Ø,Ø,Ø,56,Ø,Ø,Ø,Ø,59,Ø,Ø,Ø
9Ø OATA62,Ø,Ø,Ø,Ø,Ø,72,71,61,Ø,Ø,Ø,Ø,Ø,56,Ø,64,Ø,Ø,65,Ø,66,68,63
,Ø,Ø,Ø,64,Ø,Ø,Ø,64,67,Ø,Ø,Ø,66,Ø,Ø,Ø,Ø,71,Ø,Ø,Ø,69,Ø,64,Ø,68,Ø,Ø
,Ø,Ø,Ø,Ø,51,Ø,Ø,Ø,Ø,Ø,Ø,62,67,Ø,Ø,Ø,62,Ø,Ø,Ø
```

Program Listing 4

```
1Ø OPEN"R",1,"SFALDES:Ø"
2Ø FORX=1TO72:READOE$:FIELD1,255ASD$:LSETO$=DE$:PUT1:NEXTX
3Ø CLOSE1
1ØØ DATAYOU ARE STANDING AT AN ENTRANCE TO AN OLD ABANDONED SHAC
K. TO   THE WEST IS A ROCKY PATH. A ROAD GOES NORTH.
11Ø DATAYOU ARE INSIDE AN OLD SHACK. THERE IS A DOOR TO THE WEST
.
12Ø DATAYOU ARE STANDING AT THE EDGE OF A ROCK STREWN PATH. THE
PATH    CONTINUES EAST. SMALL WALKS LEAD TO THE NORTH AND WEST.
13Ø OATAYOU ARE IN A FOREST.
14Ø OATAYOU ARE IN A FOREST.
15Ø DATAYOU ARE IN A FOREST.
16Ø DATAYOU ARE IN THE MIDDLE OF A ROAD. THE ROAD CONTINUES TO T
HE NORTHAND SOUTH. A SMALL PATH LEADS EASTWARD.
17Ø DATAYOU ARE ON THE EDGE OF A MURKY POOL. A DARK LIQUID IS FL
OATING  ON TOP OF THE POOL. THERE IS A SMALL PATH FROM THE NORTH
. A     SMALL WALK LEADS WEST.
18Ø DATAYOU ARE IN A THREE WAY JUNCTION. A ROAD GOES NORTH AND S
OUTH.   THERE IS A PASSAGE LEADING WEST.
19Ø DATAYOU ARE AT AN ENTRANCE TO A SMALL CAVE. A SMALL NARROW H
OLE IS  WEST. A PASSAGE LEADS EAST. ON TOP OF THE HOLE A SIGN RE
ADS     `BEWARE BRAVE ADVENTURER. FOR IT IS THE SMALL THINGS IN
LIFE    THAT SO OFTEN OESTROYS IT.'
2ØØ DATAYOU ARE INSIDE A CAVE. LIGHT FILTERS THROUGH FROM THE EA
ST. A   SMALL PATH LEADS WEST. A SMALLER PATH LEADS SOUTH.
21Ø DATATHE PATH IS FULL OF BROKEN ROCKS. THE PATH CONTINUES TO
THE     NORTH AND SOUTH.
22Ø DATAYOU ARE INSIDE A SMALL CAVERN. SCATTERED THROUGHOUT THE
ROOM ARETHE REMAINS OF LESS FORTUNATE ADVENTURERS. A PATH LEADS
NORTH.  A PASSAGE LEADS WEST.
23Ø DATAYOU ARE IN A LONG HALL EXTENDING EAST ANO WEST.
24Ø DATAYOU ARE IN A ROOM LIGHTED BY A SIGN THAT SAYS `GNOMES NE
EO FEW  TOOLS TO OO THEIR DIRTY DEEOS.' PASSAGES LEAD EAST AND W
EST. A  SMALL PATH LEADS SOUTH.
25Ø OATATHIS IS THE TOOL ROOM. STREWN ALL OVER THE FLOOR ARE SMA
LL      KNIVES WHICH OISAPPEAR TO THE TOUCH. A PASSAGE IS NORTH.
 A WALK LEADS UP.
26Ø DATAYOU ARE IN A LONG CORRIDOR LEADING EAST AND WEST.
27Ø OATAYOU ARE IN A TWISTY LITTLE MAZE OF PASSAGES.
28Ø DATAYOU ARE IN A MAZE OF TWISTY LITTLE PASSAGES.
29Ø DATAYOU ARE IN A MAZE OF TWISTY LITTLE PASSAGES.
3ØØ DATAYOU ARE AT A DEAD END.
31Ø OATAYOU ARE IN A MAZE OF TWISTY LITTLE PASSAGES.
32Ø OATAYOU ARE IN A LITTLE MAZE OF TWISTY PASSAGES.
33Ø DATAA TALL WALL RISES ABOVE YOU. YOU ARE NEARLY ENCIRCLED BY
 LARGE  UNCLIMBABLE CANYONS. THE ONLY EXITS LEAD SOUTH AND EAST.

34Ø DATAYOU ARE IN A LONG HALL. ENGRAVED ON THE ROCKS IS A MESSA
GE.     EXITS LEAD EAST ANO WEST.
35Ø OATAYOU HAVE COME TO A PLACE WHERE THREE PATHS MEET. A SMALL
 PATH   LEADS NORTH. LARGER PATHS LEAD EAST AND WEST.
36Ø DATAA BROAD CHASM SEPARATES YOU FROM A LEDGE IN THE NORTH. A
 SMALL  PATH LEADS SOUTH.
37Ø DATAYOU ARE IN A SMALL NARROW LEDGE. THERE IS A CAVERN ACROS
S THE   CHASM IN THE SOUTH. A WALK LEADS DOWN.
38Ø DATAYOU ARE IN AN EXTREMELY NARROW PART OF THE CAVE. EXITS L
EAD EASTANO WEST.
39Ø DATAYOU ARE IN A `T.' A LARGE PATH LEADS EAST. A ROAD CONTIN
UES TO  THE NORTH AND TO THE SOUTH.
4ØØ DATAYOU ARE AT AN ENTRANCE TO AN OLD ABANDONED MINE. A SMALL
 PATH   LEADS SOUTH. A LARGE PATH LEADS WEST.
41Ø DATAYOU ARE STANDING BY A LARGE OAK TREE. ENGRAVED ON THE BA
RK IS A MESSAGE. PATHS LEAD NORTH ANO SOUTH.
42Ø DATAYOU ARE INSIDE A MINE. THERE ARE PASSAGES IN EVERY DIREC
TION.
43Ø OATAYOU ARE INSIDE A MINE. PATHS LEAD TO ALL DIRECTIONS.
44Ø DATAYOU ARE INSIDE A MINE. PATHS LEAD TO ALL DIRECTIONS.
45Ø DATAYOU ARE INSIDE A MINE. PATHS LEAD TO ALL DIRECTIONS.
```

Listing 4 continues

Listing 4 continued

```
460 DATAYOU ARE INSIDE A MINE. PATHS LEAD TO ALL DIRECTIONS.
470 DATAYOU ARE INSIDE A MINE. PASSAGES LEAD TO ALL DIRECTIONS.
480 DATAYOU'RE IN A ROAD. A COLD WIND BLOWS INTO YOUR FACE. IT S
EEMS TO SAY `LUCK DOESN'T RUN FOREVER.' THE ROAD CONTINUES TO TH
E NORTH AND TO THE SOUTH.
490 DATATHE ROAD GROWS SMALLER. TO THE EAST IS A SELDOM USED GRA
VEL     PATH. THE ROAD CONTINUES NORTHWARD AND SOUTHWARD.
500 DATAYOU HAVE CLIMBED UP A HILL. THE VIEW IS SPECTACULAR (AS
ONE     WOULD EXPECT). SHARP ROCKS PROTRUDE FROM THE BASE OF THE
 HILL.  BEYOND A RIVER A STRANGE HOUSE CAN BE SEEN. THE SILENCE
HERE IS UNCANNY. PATHS LEAD EAST AND WEST.
510 DATATHE GRAVEL PATH ENDS. A LARGE ROCK MARKS THIS SPOT.
520 DATAYOU ARE AT THE RIVERS' SOUTHERN EDGE. THERE IS A SMALL S
IGN ON  THE SHORE. A ROAD GOES SOUTH.
530 DATAYOU ARE AT THE RIVERS' NORTHERN EDGE. A LARGE SIGN SAYS
`WELCOMETO YOUR DEATH.' BELOW THE SIGN (SCRIBBLED IN RED) IS A M
ESSAGE. A ROAD GOES NORTH.
540 DATAYOU ARE AT A CROSSROADS. A ROAD GOES NORTH AND SOUTH. EX
ITS LEADEAST AND WEST.
550 DATATHE GROUND HERE IS LITTERED BY LARGE HOLES. THERE ARE PA
THS     LEADING EAST AND WEST.
560 DATAYOU ARE STANDING NEAR AN OLD STABLE. A SIGN READS `OLD O
DORS    NEVER DIE...THEY JUST ACCUMULATE.' PATHS LEAD NORTH AND
WEST.
570 DATATHE ROAD ENDS HERE. SMALL PATHS LEAD NORTH AND SOUTH. A
ROAD    GOES TO THE EAST.
580 DATAYOU ARE AT THE RIVERFRONT. UPON THE SHORE YOU SEE THE BO
NES     OF UNSUCCESFUL RIVER CROSSERS. A PATH LEADS NORTH.
590 DATAYOU ARE STANDING BY A LARGE GARGOYLE STATUE. IT SEEMS TO
 BE     STARING AT YOU. INSCRIBED AT ITS BASE IS THE NUMBER 13.
A ROAD  GOES SOUTH. EXITS LEAD EAST AND WEST. A HOUSE IS NORTH.
600 DATAYOU ARE OUTSIDE AN OUTHOUSE. A PATH LEADS SOUTH. OTHER P
ATHS    LEAD EAST AND WEST.
610 DATAA LARGE STONE IDOL STANDS IN FRONT OF YOU. THERE IS A SM
ALL     CHARRED PIT IN FRONT OF THE IDOL. A PATH LEADS EAST.
620 DATAYOU ARE IN A GARDEN. HALF A DOZEN FRESHLY PICKED CARROTS
 ARE    LYING ON THE GROUND. AN ORCHARD IS EAST. A PATH LEADS WE
ST.
630 DATAYOU ARE IN AN ORCHARD. SORROUNDED BY A FLOOD OF ORANGES
IS ONE  SOLITARY APPLE TREE. A PATH LEADS WEST. ANOTHER PATH GOE
S SOUTH.
640 DATAYOU ARE AT AN ENTRANCE TO AN ODD LOOKING HOUSE. IN THE M
AILBOX  IS A LETTER MARKED `C/O WIZARD OF OZ.' A WALK LEADS SOUT
H.
650 DATAYOU ARE INSIDE THE HOUSE. A LARGE DOOR IS TO THE SOUTH.
A LARGE HALLWAY CONTINUES NORTH. THERE IS AN ARCHED DOOR TO THE
EAST.   A SMALL DOORWAY IS WEST.
660 DATATHIS IS THE DINING ROOM. IT LOOKS LIKE SOMEBODY IS EXPEC
TING    GUESTS. THERE ARE SEVEN PLATES ON THE TABLE. A DOOR IS N
ORTH.   A DOORWAY IS EAST.
670 DATATHIS IS THE KITCHEN. A BIG POT OF BOILING WATER IS OVER

A WOOD  STOVE. A DOOR IS SOUTH.
680 DATAYOU ARE IN THE LIVING ROOM. A SMALL STATUE WITH A ROUND
HEAD IS SITTING ON A TABLE. A LARGE DOORWAY IS WEST.
690 DATAYOU ARE IN A SMALL HIDDEN ROOM. A ROPE HANGS FROM THE CE
ILING.  THERE IS AN OPEN PANEL TO THE SOUTH.
700 DATAYOU ARE IN THE BASEMENT. A CORRIDOR LEADS NORTH.
710 DATAA LARGE COMBINATION VAULT IS STANDING IN FRONT OF YOU. A
 SIGN ONTOP OF THE VAULT SAYS `DEPOSIT TREASURES INSIDE THE VAUL
T FOR   FULL CREDIT.' A CORRIDOR LEADS SOUTH. A DOOR IS EAST.
720 DATAYOU ARE IN THE RECEPTION HALL. A LARGE TABLE IS ON ONE S
IDE OF  THE ROOM. AS YOU GAZE UPON THE TABLE YOU SEE SOMEONE STA
RE BACK AT YOU. A CAN OF PLEDGE SITS IN THE FAR SIDE OF THE ROOM
. STAIRSLEAD UP. THE HALL EXTENDS SOUTH.
730 DATAYOU ARE IN THE MIDDLE FLOOR. STAIRS LEAD UP AND DOWN. DO
ORWAYS  ARE TO THE EAST AND WEST.
740 DATAYOU ARE IN THE LIBRARY. A LARGE HEAVY BOOK LIES OPEN ON
TOP OF  A DESK BY A WINDOW. A DOOR IS WEST.
750 DATAYOU ARE IN THE BEDROOM. A LARGE WALK IN CLOSET IS SOUTH.
 THERE  IS A LARGE DRESSER WITH A CENTRAL DRAWER NEAR THE BED. A
 DOOR   IS TO THE EAST.
760 DATAYOU ARE IN A LARGE WALK IN CLOSET WITH A DOOR TO THE NOR
TH.
770 DATAYOU ARE ON THE UPPER FLOOR. A DOOR IS WEST. STAIRS LEAD
DOWN.
780 DATAYOU ARE IN A MUSTY ATTIC. AN OLD CHEST IS SITTING IN ONE
 CORNER OF THE ROOM. A DOOR IS EAST.
790 DATAYOU ARE INSIDE THE OUTHOUSE. A STRANGE MESSAGE IS PASTED
 ON THE WALL. A DOORWAY IS SOUTH.
800 DATAYOU ARE INSIDE A SMALL CUBICLE. INSCRIBED ON TOP OF THE
DOOR ARETHE LETTERS I F AND T. ONE OTHER LETTER APPEARS TO BE MI
SSING.  THE DOOR IS WEST.
810 DATAYOU ARE INSIDE A VAULT.
```

Program Listing 5

```
10 OPEN"R",1,"SFASDES:0":FIELD1,255ASDE$
20 FORX=1TO72:READD$:LSETDE$=D$:PUT1:NEXT:CLOSE1
25 DATAYOU ARE OUTSIDE AN OLD SHACK.,YOU ARE INSIDE A SHACK.,YOU
 ARE ON THE EDGE OF A LARGE PATH.
30 DATAYOU ARE IN A FOREST.,YOU ARE IN A FOREST.,YOU ARE IN A FO
REST.,YOU'RE IN THE MIDDLE OF A ROAD.,YOU ARE BY A MURKY POOL.,Y
OU'RE IN A THREE WAY JUNCTION.,YOU'RE AT THE ENTANCE TO A CAVE.
40 DATAYOU'RE INSIDE PLACE WHERE LIGHT FILTERS FROM THE EAST.,YO
U ARE IN A PASSAGE OF BROKEN ROCKS.,YOU'RE INSIDE A SMALL CAVERN
.,YOU ARE IN A LONG HALL.,THIS IS A ROOM LIGHTED BY A SIGN.,YOU'
RE IN THE TOOL ROOM.,YOU ARE IN A LONG CORRIDOR.
50 DATAYOU ARE IN A TWISTY LITTLE MAZE OF PASSAGES.,YOU ARE IN A
 MAZE OF TWISTY LITTLE PASSAGES.,YOU ARE IN A MAZE OF TWISTY LIT
TLE PASSAGES.,YOU ARE AT A DEAD END.,YOU ARE IN A MAZE OF TWISTY
 LITTLE PASSAGES.
60 DATAYOU ARE IN A LITTLE MAZE OF TWISTY PASSAGES.,YOU'RE IN TH
E TALL WALL CANYON.,YOU'RE IN A LONG HALL.,YOU ARE IN A PLACE WH
ERE THREE PATHS MEET.,YOU'RE AT THE CHASM.,YOU ARE IN A NARROW L
EDGE.,YOU'RE IN NARROW PART OF CAVE.
70 DATAYOU ARE IN THE `T.',YOU ARE AT THE MINE ENTRANCE.,YOU ARE
 NEAR AN OAK TREE.,YOU ARE INSIDE THE MINE. PASSAGES LEAD TO ALL
 DIRECTIONS.,YOU ARE INSIDE A MINE. PATHS LEAD TO ALL DIRECTIONS
.
80 DATAYOU ARE INSIDE A MINE. PATHS LEAD TO ALL DIRECTIONS.,YOU
ARE INSIDE A MINE. PATHS LEAD TO ALL DIRECTIONS.,YOU ARE INSIDE
A MINE. PATHS LEAD TO ALL DIRECTIONS.,YOU ARE INSIDE A MINE. THE
RE ARE PASSAGES IN ALL DIRECTIONS.
90 DATAYOU ARE IN A PLACE WHERE COLD WINDS BLOW.,THE ROAD NARROW
S HERE.,YOU'RE ON TOP OF A HILL.,YOU ARE BY A LARGE ROCK.,YOU AR
```

Listing 5 continues

RUN BASIC PROGRAMS AT SUPER SPEED WITH ZBASIC 2.2.

THE WORLDS FASTEST TRS-80 BASIC COMPILER from SIMUTEK

BELIEVE IT OR NOT WE'VE ADDED MORE NEW FEATURES to the ONLY INTERACTIVE BASIC COMPILER for the TRS-80!

1. Speed increases of 10–100 times are typical after compilation.
2. Compiled code can be RELOCATED to run anywhere in memory. Code is even ROMable!
3. ZBASIC 2.2 NOW SUPPORTS BOTH RANDOM and SEQUENTIAL DISK I/O.
4. ZBASIC 2.2 is now a super tool for business programmers: RANDOM ACCESS FILES, and PRINT USING statements are supported as well as a HIGH PRECISION MATH package (with no rounding problems).
5. Special BUILT-IN MACHINE LANGUAGE COMMANDS to increase program operation by as much as 1000 times! Special commands are implemented for fast memory searching (CPDR, CPIR), block memory moves (LDIR, LDDR), inputting and printing HEX numbers, inserting MACHINE LANGUAGE into COMPILED CODE, disabling and enabling interrupts, inverting memory, 16 bit PEEKs and POKEs, and stack control, debug and much more.
6. ZBASIC 2.2 compiles the ENTIRE PROGRAM into to Z-80 machine language. (Not 8080 code or a combination of BASIC and machine language like some other compilers.) Clumsy LINKING LOADERS, and RUNTIME MODULES are not needed; ZBASIC 2.2 creates a ready to run MACHINE LANGUAGE program.
7. NO ROYALTIES imposed on registered ZBASIC owners.
8. Typical COMPILATION TIME is TWO SECONDS for a 4K program.
9. Use TRS-80 Basic to write ZBASIC programs!
10. Compile many existing programs with only minor changes. (Some BASIC programming experience is required.)
11. Fully compatible with both the Model I and the Model III. Mod I compiled programs work on a MODEL III, and visa-versa. ZBASIC works with NEWDOS-80, NEWDOS+, DOSPLUS, LDOS, MULTIDOS, ULTRADOS, TRSDOS etc.
12. BUILT-IN and much improved MUSIC and SOUND EFFECTS commands.
13. Improved CHAINING for disk users.
14. TIME$ now available on DISK version.
15. ZBASIC 2.2 now has an INPUT @ command (similar to PRINT @).
16. The TAB function will now tab 255 columns on a printer. (BASIC cannot tab past column 64.)
17. NEWDOS 80 2.0 USERS can use the CMD "dos command" function!
18. NEW and EASIER to use USR COMMANDS.
19. New math functions to calculate XOR and INTEGER REMAINDERS
20. Logical STRING COMPARISONS are now supported.
21. The disk commands INSTR, MID$ ASSIGNMENT are now supported on both DISK AND TAPE ZBASIC.
22. DEFSTR is now supported.
23. Eight disk files may be opened simultaneously; random, sequential or mixed.
24. LINE INPUT#, is now supported
25. Invoke the compiler by simply hitting these two keys: ":-"
26. NEW 100+ PAGE MANUAL WITH DESCRIPTIONS AND EXAMPLE.
27. ZBASIC 2.2 Comes with CMDFILE/CMD program from MISOSYS, to allow appending or merging compiled programs and machine language programs from tape or disk.

ZBASIC 2.2 DOES NOT SUPPORT THESE BASIC COMMANDS:

1. ATN, EXP, COS, SIN, LOG, TAN, and exponentiation. (However, subroutines are included in the manual for these functions.)
2. ERROR, ON ERROR GOTO, ERL, ERR RESUME.
3. No direct commands like AUTO, EDIT, LIST, LLIST ETC, although these commands may be used when writing programs.
4. Others NOT supported: CDBL, CINT, CSNG, DEFFN, FIX, FRE.
5. Normal CASSETTE I/O. (ZBASIC supports it's own SPECIAL CASSETTE I/O statements.)
6. SOME BASIC COMMANDS MAY DIFFER IN ZBASIC. For instance, END jumps to DOS READY, STOP jumps to BASIC READY etc.
7. MEMORY REQUIREMENTS: to approximate the largest BASIC program that can be compiled in your machine (at one time), enter BASIC and type: PRINT (MEM-6500)/2. Remember, you can merge compiled programs together to fill memory.

ZBASIC 2.2 SPEED COMPARISON DEMO

To help give you an idea how fast compiled programs are, we have included this demo program:

ZBASIC 2.2 DEMO PROGRAM

Time to compile and run complete program	**: 0 MIN. 2 SEC.**
BASIC Execution speed MOD I, LEVEL II	**: 7 MIN. 34 SEC.**
ZBASIC Execution speed MOD I, LEVEL II	**: 0 MIN. 18 SEC.**
BASIC Program size (WITHOUT VARIABLES)	**: 895 BYTES**
ZBASIC Program size (WITHOUT VARIABLES)	**: 2733 BYTES**

(Remember that the ZBASIC program includes an 1879 byte subroutine package.) Program shown exactly as compiled and run in BASIC and ZBASIC.

```
10 '========= ZBASIC 2.2 EXAMPLE PROGRAM AND TIME TEST========
20 CLS:CLEAR100:DEFINT A-X:DEFSTR Z:DIM AA(64,24),Z(50):RANDOM
30 AA=100:BB=-1000:CC=3:DD=-3:EE=-9999:ST$="START TIME "+TIME$
40 FOR I=1TO127STEP2 :FOR J=47TO1STEP-3:XX=POINT(I,J):SET(I,J)
50 XX=(I-J)/CC*(7+I+J) :XX=ABS(INT(RND(I*J)-AA)+7) :RESET(I,J)
60 XX=PEEK(I+J) :POKE15360+I+J,J :OUT255,J AND (3*J):XX=INP(I)
70 AB$=STR$(I+J) :BA$=LEFT$(AB$,2) :AA(I/2,J/2)=VAL(BA$)+AA*3
80 BA$=BA$+RIGHT$(BA$,RND(3)) :XX=INSTR(I,BA$,"9"):XX=SQR(I*J)
90 BA$=MID$(BA$,2,2) :MID$(BA$,1,1)=Z :IF XX THEN 100 ELSE CLS
100 IF LEN(BA$))3 OR SGN(XX)=1 AND ASC(BA$)=32 THEN PRINT"+++";
110 IFPOS(0))62 THEN TRON:TROFF:PRINT ELSE XX=NOT(RND(99))+100
120 A$=INKEY$:IF A$="Y" OR A$="y" AND I)120 THEN PRINT"TRUE.."
130 RESTORE :READA,C,Z(J),D:GOSUB170:GOSUB170:GOSUB170:GOTO210
140 NEXT :PRINT"*";:NEXTI:CLS:PRINT@512,ST$,"STOP TIME ";TIME$
150 STOP'=============== END OF MAIN TEST LOOP =================
160 DATA 12345,-1,"TEST",-9999
170 ON RND(6) GOTO 180,190,200,180,190,200
180 RETURN
190 RETURN
200 RETURN
210 ON RND(9) GOSUB 180,190,200,180,190,200,180,190,200
220 GOTO140
```

NOTICE ZBASIC 2.0 OWNERS: you can upgrade your ZBASIC 2.0 for no charge. Just send us your original diskette/cassette and a S.A.S.E with your registered serial number and copy of your invoice. We will send you ZBASIC 2.2 and updates to your manual.

VISA, MASTERCARD, AMERICAN EXPRESS, C.O.D. ORDERS ONLY.

800 528-1149 order line

ZBASIC 2.2 DISK VERSION AND MANUAL	89.95
ZBASIC 2.2 TAPE VERSION AND MANUAL	79.95
ZBASIC 2.2 DISK & TAPE VERSION AND MANUAL	99.99
MANUAL ONLY... (APPLIES TO PURCHASE)	25.00

SIMUTEK COMPUTER PRODUCTS INC.

TECHNICAL QUESTIONS PLEASE CALL (602) 323-9391
4897 E. SPEEDWAY, TUCSON, ARIZONA 85712 ✓12

TRS-80 is tm of Radio Shack, a Tandy Corp.

✓See List of Advertisers on Page 563

Listing 5 continued

```
E AT THE RIVER'S SOUTHERN EDGE.,YOU ARE AT THE RIVER'S NORTHERN
EOGE.,YOU'RE AT A CROSSROADS.
100 OATAYOU ARE IN `HOLEY' GROUND.,YOU'RE NEAR THE STABLE.,THE R
OAD ENDS HERE.,YOU ARE AT THE RIVERFRONT.,YOU ARE BY A GARGOYLE
STATUE.,YOU'RE OUTSIOE AN OUTHOUSE.,A LARGE STONE IOOL STANOS IN
 FRONT OF YOU.,YOU'RE IN A GAROEN.
110 DATAYOU'RE IN AN ORCHARO.,YOU ARE STANOING AT AN ENTRANCE TO
 A HOUSE.,YOU'RE NEAR THREE DOORWAYS.,THIS IS THE OINING ROOM.,Y
OU'RE IN THE KITCHEN.,YOU ARE IN A LIVING ROOM.,YOU ARE INSIDE A
 SMALL ROOM.,YOU ARE IN THE BASEMENT.
120 DATAYOU'RE OUTSIDE A VAULT.,THIS IS THE RECEPTION HALL.,YOU
ARE IN THE MIDDLE FLOOR.,YOU ARE IN THE LIBRARY.,YOU ARE IN THE
BEORROOM.,YOU ARE INSIOE A WALK-IN CLOSET.,THIS IS THE TOP FLOOR
.
130 DATAYOU ARE IN THE ATTIC.,YOU'RE INSIDE AN OUTHOUSE.,YOU ARE
 INSIDE A SMALL CUBICLE.,YOU ARE INSIDE A VAULT.
```

Program Listing 6

```
10 OPEN"O",1,"SFAVOC:0"
20 FORX=0TO96:READV$,V:PRINT#1,V$","V:NEXT:CLOSE1:ENO
100 OATANORT,1,N,1,EAST,2,E,2,SOUT,3,S,3,WEST,4,W,4,UP,5,U,5,OOW
N,6,D,6,LOOK,7,HINT,8,HELP,8,INVE,9,JUMP,10,SCOR,11,QUIT,12,SAVE
,13,REST,14,ENTE,15,LEAV,16,TAKE,17,GET,17,T,17,OBTA,17,OROP,18,
DR,18,WAVE,19,SHOW,20,READ,21,FILL,22
110 DATALIGH,23,BURN,23,KILL,24,ATTA,24,KICK,25,SMAS,26,BREA,26,
THRO,27,TOSS,27,EAT,28,DRIN,29,FEED,30,TURN,31,UNLO,32,CROS,33,S
WIM,34,OPEN,35,CLOS,36,OFF,37,PULL,38,EMPT,39
120 DATADIAM,0,EMER,1,CHAI,2,PLAT,2,APPL,3,GOLO,3,PEAR,4,SPIC,5,
OAGG,6,KNIF,6,FIGU,7,LAMP,8,KEY,9,RING,10,ROO,11,STAF,12,INCE,13
,HAY,14,STRA,14,CHEE,15,MOUS,16,MATC,17,BOTT,18,IOOL,19,SNAK,20,
ABYS,21,CHAS,21,WIZA,22,RIVE,23,HEAD,24,BOOK,25
130 DATAMESS,26,BALL,27,CRYS,27,OIL,28,WATE,29,VAUL,30,CHES,31,O
RAW,32,WIND,33,ROPE,34,SIGN,35,GNOM,36
```

Program Listing 7

```
5 OPEN"O",1,"SFAOBOES:0"
10 FORX=0TO29:READOB$:PRINT#1,X","OB$:NEXT
30 FORX=0TO19:READOB$:PRINT#1,X","OB$:NEXT
40 FORX=0TO28:READOP:PRINT#1,OP",";:NEXTX:PRINT#1,0:CLOSE1:ENO
100 OATAA LARGE DIAMONO IS LYING HERE.,THERE'S AN EMERALD EYE HE
RE.,A PLATINUM CHAIN IS LYING HERE.,THERE IS A GOLDEN APPLE HERE
.,A STRING OF PEARLS IS LYING HERE.,THERE IS A BOTTLE OF RARE SP
ICES LYING HERE.
110 DATATHERE IS A JEWEL ENCRUSTED DAGGER IN HERE.,A PRICELESS F
IGURINE IS SITTING ON SOME HAY.,THERE IS AN OLO OIL LAMP HERE.,A
N OLD BRASS KEY IS SITTING HERE.,THERE IS A WORTHLESS LOOKING RI
NG HERE.,AN OLO BLACK ROD IS SITTING HERE.
120 OATATHERE'S AN ANCIENT LOOKING STAFF LYING HERE.,THERE'S A P
```

Listing 7 continues

```
ACKAGE OF INCENSE LYING HERE.,THERE'S A BALE OF HAY HERE.,A PACK
AGE OF KRAFT CHEESE IS LYING HERE.,THERE IS A NOISY LITTLE MOUSE
 HERE.,THERE'S A BOX OF MATCHES LYING HERE.
130 OATATHERE'S AN EMPTY BOTTLE HERE.,THERE'S A BROWNISH BOTTLE
OF WATER HERE.,AN EMERALD EYE SITS ON TOP OF THE IDOL.,THERE IS
A KNIFE WIELOING GNOME IN THE ROOM WITH YOU.,THERE ARE RARE SPIC
ES SITTING ON THE GROUND.
140 OATAAN ANGRY WIZARD WITH A PLATINUM CHAIN IS IN THE ROOM WIT
H YOU.,A GIANT SNAKE IS JEALOUSLY GUARDING A GOLOEN APPLE.,THERE
 IS A LARGE CRYSTAL BALL ON THE GROUND.,A PRICELESS FIGURINE STA
NOS ON TOP OF THE TABLE.
145 OATATHE WINDOW IS OPEN.,THE WINDOW IS CLOSED.,THERE IS AN OP
EN PANEL NORTH.
150 OATALARGE DIAMONO,EMERALD EYE,PLATINUM CHAIN,GOLDEN APPLE,ST
RING OF PEARLS,BOTTLE OF SPICES,JEWELED DAGGER,PRICELESS FIGURIN
E,OIL LAMP,BRASS KEY,RING,BLACK ROD,WOODEN STAFF,INCENSE,HAY,KRA
FT CHEESE,NOISY MOUSE
160 DATAMATCHES,EMPTY BOTTLE,BOTTLE OF WATER
170 DATA38,0,0,0,0,0,15,0,2,2,49,21,42,58,47,57,69,6,24,0,52,0,2
8,61,54,32,63,65,0
```

Program Listing 8

```
5 OPEN"R",1,"SFARESP:0"
10 FIELO1,255ASR$
15 FORX=1TO102:READRE$:LSETR$=RE$:PUT1:NEXT:CLOSE1
100 OATATHERE IS NO WAY TO GO IN THAT OIRECTION.,I DON'T UNDERST
ANO.,OK.,I DON'T SEE IT HERE.,YOU'RE NOT CARRYING IT.,NOTHING HA
PPENS.,YOU'RE ALREADY CARRYING IT!,IN A BLAZE OF GLORY YOU FINO
YOURSELF IN A LANO FAR AWAY.,IT'S TOO HEAVY FOR YOU TO TAKE.
110 OATAOUCH! EVERY BONE IN YOUR FOOT JUST BROKE.,EECH! YOUR HAN
DS JUST TURNED INTO A BLOODY MESS.,YOU DON'T HAVE ANY MATCHES.,Y
OU CANNOT LIGHT AN EMPTY OIL LAMP.,YOU HAVE NO KEYS.
120 OATATHERE IS NOT ENOUGH LIGHT TO SEE AROUND YOU.,YOU HAD BET
TER HURRY. TIME IS RUNNING OUT.,YOU'RE LAMP IS GETTING OIM.,THE
MOUSE IS FRIGHTENEO BY YOU.,THE ONLY THING EDIBLE HERE IS YOU!,T
HE MOUSE WAS USEO FOR CARCINOGENIC TESTING. THE SNAKE IS OEAD.
130 OATATHE SNAKE SEVERS YOUR HANO. YOU HAVE BEEN POISONED.,THE
RING GLOWS BRIGHTLY. A LIGHTNING BOLT STRIKES THE WIZARD.,THE GO
D IS INCENSEO BY YOUR AUDACITY. WITH A BLINK OF AN EYE    YOU FA
LL DEAD TO THE GROUNO.,AN EMERALD EYE FALLS TO THE GROUND.
140 DATAA PLATINUM CHAIN LIES BEFORE YOU.,A GOLOEN APPLE HANGS B
Y AN APPLE TREE.,YOU CAN'T CROSS THE RIVER.,HOLY MOSES! THE RIVE
R JUST SPLIT IN TWO! A ORY PATH LEADS TO    THE OTHER SIDE.,A LO
UO NOISE SORROUNDS YOU AS THE RIVER CLOSES.
150 OATAYOU OIO NOT JUMP LONG ENOUGH. YOU FELL INTO THE RIVER.,A
 SCHOOL OF PHYRANIAS DEVOURS YOU.,FISH FROM THE AMAZON.,FRODO LI
VES.,MERLIN WAS HERE., A CRYSTAL BRIDGE NOW SPANS THE CHASM!,THE
 BRIDGE HAS JUST AS SUODENLY DISAPPERARED!
160 DATATHE GREAT ALMAZAR BID'S YOU WELL. THOUGH YOU WILL ENCOUN
TER MANYTRIALS HE SHALL PROVIOE FOR YOU. HE THAT IS BOTH WATER A
NO FLAMESHALL SENO YOU A GIFT TO AIO YOU IN YOUR QUEST. THUS SAY
ETH     ALMAZAR `LIVE ANO YOU SHALL LIVE.'
170 DATA`ANO SO IT CAME TO PAST THAT IN THE THIRD DAY OF THE SEV
ENTH     MONTH OF THE TWENTY-FIRST YEAR A GREAT EVENT HAPPENED. T
```

Listing 8 continues

Listing 8 continued

```
HE EARTHOPENED AND TOOK WITHIN IT THE FRUITS OF THE PEOPLES' LAB
OR AND  GAVE UPON THEM A GREATER GLORY.'
180 DATATHE CHEST IS EMPTY.,WHEW! THAT'S STRONG STUFF!,YUCK! THE
 MOUSE WAS USED FOR CARCINOGENIC TESTING. YOU'RE DEAD.,THE ABYSS
 IS TOO WIDE TO CROSS.,YOU FELL INTO THE CHASM. YOU ARE DEAD.,A
SMALL AXE FLIES THROUGH THE WINDOW AND STRIKES YOU DEAD.
190 DATATHE WIZARD RAISES HIS HAND AND IN AN INSTANT YOU FIND YO
URSELF  UNCOMFORTABLY HOT. AS YOU LOOK UP INTO THE SKY YOU SEE A
 LARGE  STONE IDOL. YOU ARE BEING OFFERED AS A SACRIFICE.
200 DATAEVERYTHING SPINS AROUND YOU AS A WALL OF FIRE ENCIRCLES
YOU.    WHEN YOU WAKE UP YOU FIND THAT...,A SMALL WALL OF FIRE B
ARELY SORROUNDS YOU. YOU REGAIN          CONSCIOUSNESS AND FIND
 THAT...
210 DATATHAT JUMP WOULD HAVE WON A GOLD MEDAL. UNFORTUNATELY YOU
 LANDED HEAD FIRST INTO A PILE OF ROCKS.,YOU HAVE DIED THREE TIM
ES. THE GAME IS THEREFORE SUSPENDED.
220 DATAYOU OBVIOUSLY HAVE NO IDEA OF WHAT'S HAPPENING IN THIS G
AME.,YOU ARE OBVIOUSLY A RANK NOVICE IN THIS GAME.,YOU MAY NOW C
ONSIDER YOURSELF AN INTERMEDIETE IN THIS GAME.,YOU ARE NOW AN AD
VANCED ADVENTURER.
230 DATAALL HONOR THEE - GRANDMASTER ADVENTURER!,A LOUD VOICE EM
ANATES FROM BEYOND SAYING `TIME GROWS SHORT       - MORTAL -  BEF
ORE WE FEAST AGAIN.',EECH! WHAT LOUSY TASTING WATER.,THERE'S NO
WATER WITHIN REACH.
240 DATAYOU CAN'T ENTER ANYTHING FROM HERE.,YOU CANNOT LEAVE FRO
M HERE.,YOU HAVE NO CONTAINER TO PUT IT IN.,THE SNAKE BLOCKS YOU
R WAY.,YOUR CONTAINER IS FULL.,YOU DON'T HAVE A LAMP.,THE WIZARD
 BLOCKS YOUR WAY.
250 DATAYOU INCOMPETENT FOOL! YOU FORGOT TO CLOSE THE RIVER AND
AS A    RESULT A FLOOD HAS OVERRUN THE COUNTRYSIDE.,THE MOUSE RU
NS AWAY.,DISGRESSION WILL PREVENT TRANSGRESSION.
260 DATAIT'S ALREADY OPEN.,IT'S ALREADY CLOSED.,A GIANT OGRE GRA
BS YOU AND THROWS YOU INTO A PIT.,THERE IS NO SIGN OF ANY OIL HE
RE.,A PANEL SLIDES OPEN REVEALING A PASSAGE TO THE NORTH.
270 DATAA SUDDEN WIND BLOWS FROM BEHIND AND SCATTERS THE SPICES
TO THE  FOUR CORNERS OF THE EARTH.
280 DATATHE HEAVENS PART AT THEIR SEAMS AS THE GODS PREPARE THEI
R NEXT  MEAL. ALMAZAR CAN NO LONGER PROTECT YOU. FOR EVEN THE MI
GHTIEST OF BEINGS CANNOT ALTER THAT WHICH IS PREORDAINED. YOUR L
IFE IS  NOW OVER. MAY YOU REST IN PEACE.
290 DATATHE DELICATE FIGURINE BREAKS UPON HITTING THE GROUND.,A
STRING OF PEARLS IS LYING ON THE DRAWER.,YOU'RE NOT CARRYING A B
OTTLE.,THERE IS NOTHING TO FILL IT WITH.
300 DATAA TRAP DOOR OPENS BENEATH YOU. THE FALL KNOCKS YOU UNCON
SCIOUS. WHEN YOU AWAKEN YOU FIND THAT...,THE CHEESE WAS STALE BU
T YOU MANAGED TO CONSUME THE ENTIRE BOX.
310 DATATHE GNOME IS TOO FAST FOR YOU.,THE LAMP IS ALREADY LIT.,
YOU'VE GOT TOO MUCH TO CARRY.,THE BOTTLE BREAKS ON IMPACT.,THE P
ANEL CLOSES AND SEALS THE PASSAGE.
320 DATAIT'S LOCKED.,YOU HAVE MANAGED TO UNLOCK THE VAULT.,THE V
AULT IS LOCKED.,THAT COMBINATION IS INCORRECT.,YOUR BOTTLE DOES
NOT CONTAIN ANYTHING.,YOU HAVE TO OPEN THE VAULT TO GET INSIDE I
T.,YOUR LAMP IS ALREADY OFF.,THE WIZARD WON'T LET YOU.
330 DATAHAY IS SOFT.,THE FIGURINE IS EXTREMELY DELICATE.,SNAKES
LOVE ROODENTS.,ONLY A POWER STRONGER THAN THE WIZARDS' OWN CAN DE
FEAT HIM.,COMPLETE YOUR TASK AND RETURN HERE AND...,THE HUNGRY I
S ALWAYS RESTLESS.
340 DATAGODS ARE UNHAPPY CREATURES WITH MANY NEEDS.,THE RIVER FL
OWS TOWARDS THE RED SEA.,THERE IS NO HELP TO BE OBTAINED.
```

Random Access

by Karl L. Townsend

Joe Mulligan's tennis score is somewhere on the HighNet Tennis Club's disk. Can you find the score without reading the entire disk?

So you have just installed disks! Your cassette player is tucked away in the top of the closet, and with it those 15-minute waits for loads or saves. Even better, with the disk's random access ability you can read any single record—just jump into the middle of the file and get it.

Get to Work!

The HighNet Tennis Club scores now reside on a new disk. Joe Mulligan's score record needs to be updated. That's easy—just pull it from the file into memory, add the new score and write it back out. But where is it? How do you get the record when you don't know where it is?

When the file was transferred from cassette to disk, entries were in alphabetical order by name. They should still be in the same order. This means Joe's data is somewhere near the middle of the file. This isn't much help—the program is looking for a record number to find the data.

Well, you could just set up a loop, read each record in alphabetical order until Joe's record is found, change it and rewrite it.

Except for faster access from the disk, this sounds suspiciously like sequential operation used with a cassette. The question remains: How do we find a specific record in a random access file?

Strategies

Since the tennis file is sorted by names in alphabetical order, several possibilities exist. As in the sample, a simple sequential read and compare will work. It is the same procedure used in any sequential file, but loses much of the advantage of random access. (You can still write the record randomly if you know where you want to write.)

Also, it's time consuming. Using a 100-record file, a desired record might be found anywhere from record 1 to record 100. Therefore, our best case (quickest find) is read one record. This procedure is good for Andy Aaron, but what about Sam Zwick? It looks like the whole file will have to be read to look at Sam's record. This is a worst case (slowest find) of about 100 read/compares. If we assume a random distribution, we will have an average of 50 read/compares. Imagine if this file had 10,000 records—that means 5,000 read/compares on the average!

A second approach is simply to list the file with record numbers. Then we just check the list for Joe Mulligan, note the record number and get it. But suppose the file is changed—a new listing is required each time. This might not be too bad with a small file, but remember that 10,000-record file.

I use a manual filing system for file folders. When I pull a file from the file

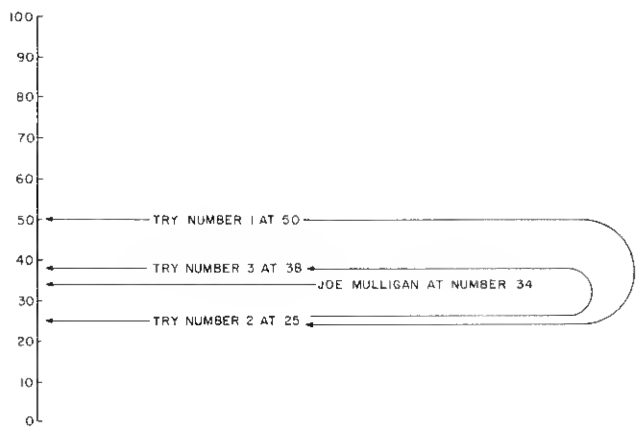


Fig. 1. Search Steps in the Halving Method

The Key Box

Model I or III
16K RAM
Cassette Basic

cabinet I always replace it in the very front of the drawer. After a time, the front of the drawer contains the folders I use most often and the least used move to the back. Every once in a while I remove those in the back and place them in dead storage.

You can adapt this for the computer. Each time a record is accessed, instead of replacing it where it was found, add it at the end of the file. (It's easier to add at the end than at the front.) When you are looking for a specific record, start reading from the end of the file. You should find it sooner this way if there is any difference in how often you access records. (The most often accessed will migrate toward the end of the file.)

Your worst case remains the maximum number of records in the file. Average accesses required to find your selected record is difficult to determine. If there is a difference in how often an individual record is accessed, it should be less than 50 in a 100-record file. However, if there is no difference in frequency of record use, the average can approach the worst-case condition as those requiring next access gravitate to the front of the file (last to be read). If file size grows with each read/write and the file eventually requires purging, it looks like this approach is loaded with too many problems to be practical.

Tries	Finds	Binary
1	102	1
2	199	2
3	392	4
4	820	8
5	1526	16
6	3210	32
7	6427	64
8	12633	128
9	25484	256
10	51017	512
11	103092	1024
12	198617	2048
13	202975	----
14	393506	----

Fig. 2. The number of read/compares to find a random record in a 10,000-record file. Notice the similarity to a binary progression.

Let's take another look at the sequential read/compare previously discussed. The major problem is the number of reads required to find an entry. If we could reduce these numbers, a read/compare still might be the way to go.

Instead of reading from the start of the file, why not start in the middle? Since the file is arranged alphabetically, we can simply read backward or forward from the center depending on whether the compare is greater or less than the record being sought. Our worst case will now be one-half the file, or 50 read/compares for our 100-record file. The average will decrease from 50 to 25, a 2-1 gain. This is better, but maybe we can carry the idea further.

Suppose, instead of starting to read sequentially when we make our compare on the middle record, we go to the middle of that half the record is in. A read/compare here tells which quarter of the file the record is in. We are now down to a maximum of 25 records to read. Actually, we can continue halving each remainder until we find the record. Confused? Take a look at the illustration in Fig. 1.

In this alphabetically sorted file of 100 records, we are looking for Joe Mulligan, whose data is in record number 34. Of course, we don't know this number or we could simply call the record by number.

Reading record number 50 and comparing the name found there with Mulligan, we find we are too far into the file; Joe's record must be in the first half. We now read record 25 and compare. This time we find we are too early in the file and that Mulligan must be in the second quarter (record 26 to 49). Again, we go to the center of the section (in this case, record number 38). We are now too far into the file and must move back. Keep halving the area and looking at the center record of each half and eventually we will find the record sought.

What do we save? I made tests on a file of 10,000 records. One million iterations were run of random selections from this file with rather interesting results. In every case the desired record was found by the fourteenth read—and this on a 10,000-record file! Compare this worst case of 14 with the 50 average reads required on a 100-record file using straight sequential methods.

Typically a find was made by 12 or 13 reads. Smaller files require fewer reads (100 records use up to seven reads) but as you can see, you will not save much nor will large files require many more reads.

For the statisticians in the group, Fig. 2 shows the results of the one million test runs. Note the number of finds made in each read and compare it to the finds preceding and following. Each time a read is made, the file remaining is cut in half and the number of finds appears to double.

Like to make this test yourself? The program listing demonstrates this search method. R defines the file size (in this case, 1,000 records). The program will loop through 100 successive finds of randomly selected records. A count of the tries in each test is saved in array G, printed at the conclusion of the run. You can change the file size by setting R to different values and the number of iterations by changing the loop value for K in line 100. Don't try a million unless you have a lot of time. Of course, change the LPRINTs to Prints if you have no printer. Type it in and give it a try. ■

```
5 '  FILE SEARCH TEST PROGRAM - KARL L. TOWNSEND - 30 JUNE 1981
10 DEFINT A-Z: DIM G(15): RANDOM 'HOUSEKEEPING
20 R = 1000                      'SET FILE SIZE
30 FOR K = 1 TO 100              'SET ITERATION LOOP
40 S = RND(1000)                 'GIVES RANDOM RECORD TO FIND
60 T = INT((R/2)+.5)             'FIND MIDDLE OF FILE TO START
70 T2 = T                        'SAVE HALVING RESULT
80 FOR I = 1 TO 15               'LOOP CHECKING SELECTION TO FILE
85 Q = I                         'SAVES LOOP CNT FOR NUMBER OF TRYS
90 T = INT((T/2)+.5)             'RE-HALVES FILE
100 LPRINT USING "#####";T2;     'PRINT RECORD NUMBERS TRIED
110 IF T2 = S THEN GOTO 151      'FIND IS MADE
120 IF T2 > S THEN T2 = T2 - T ELSE T2 = T2 + T  'FIND PROPER F
LE HALF
140 NEXT I
151 LPRINT
160 G(Q) = G(Q) + 1              'ACCUMULATE COUNT OF FINDS
170 NEXT K                       'GO BACK AND MAKE ANOTHER TEST
175 PRINT
180 FOR J = 1 TO 15
190 LPRINT G(J);                 'PRINT SUMMARY OF FINDS
200 NEXT J
210 LPRINT
220 END
```

Program Listing

Auto-Dial/Auto-Answer

by Alan Moyer

Do you ever wish that your computer could automatically dial and answer your phone? With this addition to your modem, it can.

When I bought my modem, I wanted it to have an auto-dial/auto-answer feature. But since I couldn't justify the extra expense, I settled for a basic modem—a Novation D-Cat (direct-connect). After working with the D-Cat for a while, I decided to build an interface that would add the auto-dial/auto-answer feature. I wanted my interface to be:

- easy to design and build,
- independent from the telephone line's power so it wouldn't interfere with normal phone operation,
- designed to be plugged into the expansion bus of my computer, and
- inexpensive.

Figure 1 shows the interface I designed. It consists of six integrated circuits, two 12-volt relays, two diodes, two resistors, and one magnetic reed switch, all available from Radio Shack.

The integrated circuits IC1, IC2, and IC3 operate Z80 I/O port number 254 (FE hex) to allow the computer to read and write to that port. When this port is read from, the gates in IC4 are opened so the ring-detect signal and the hook-sense signal can pass through. IC5 is a flip-flop that sets the phone either on-hook or off-hook, and operates the dialing circuit. Being a flip-flop, once it is set on or off, it stays there until the state is deliberately altered. All the signals from the computer are standard from the expansion bus.

This is not a fancy circuit, and it is similar to one shown in the *TRS-80 Technical Reference Manual,* page 89, which I used as a guide.

This interface circuit hooks to the internal circuit block of the phone as shown in Fig. 2. Except where noted, all connections are made directly to the terminals. Lines C and J require that the existing wires from the terminal block be removed and hooked directly to these lines. In effect, relay 1 parallels the internal switches inside the phone. Relay 2 is connected in series with the internal dialing circuit. It is currently hooked up to a touch-tone phone, but uses normal pulse dialing. Because of this arrangement, this method of dialing and hook control does not require power from the phone.

To make the modem detect a ringing phone, I taped a magnetic reed switch to the side of the phone's ringer coil. When the ringer coil rings the bell, the switch closes with it. Although it does cycle on and off with the coil's magnetic field, the first closing of the switch is all

The Key Box

Model I
32K RAM
Disk Basic
1 Disk Drive, RS-232, modem, LDOS

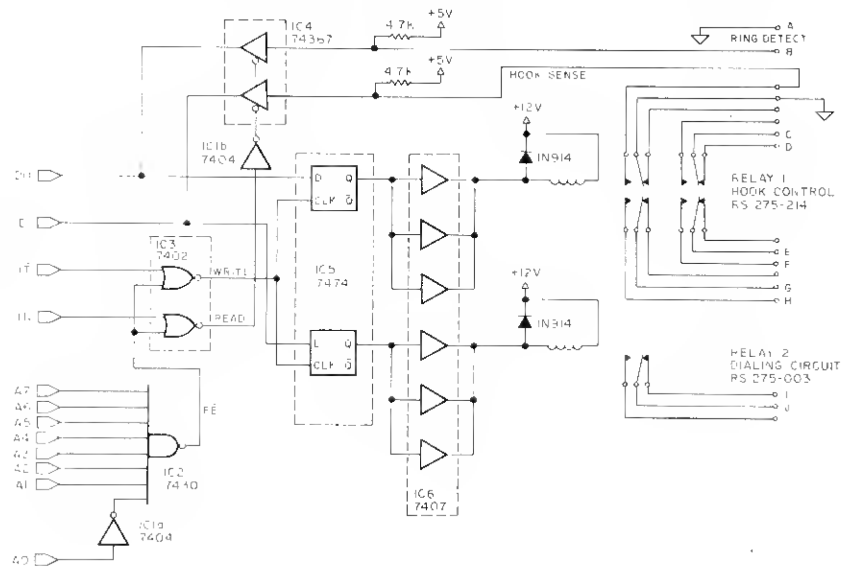


Figure 1

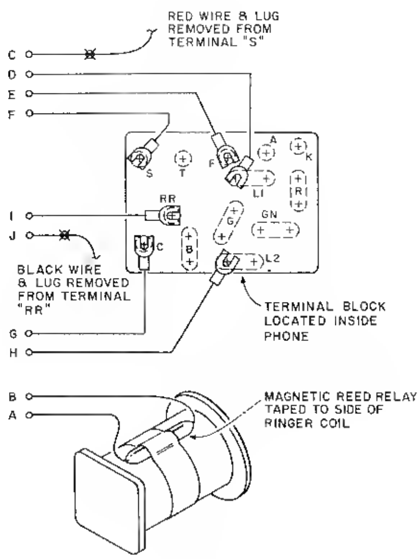


Figure 2

we need to use, so it need not be altered. In addition, this idea met all my goals for an interface.

Software

With the circuit using Z80 port FE hex (254 decimal) to access the device, Basic's command A = INP(254) will return A with the following bit information:

```
0000 0000
        ! !
        ! !—Bit zero, ring detect.
        !       1 = no ring (power-up
                condition).
        !       0 = ring.
        !
        !—      Bit one, hook sense.
                1 = on hook (power-up
                condition).
                0 = off hook.
```

And accordingly, Basic's command OUT 254,A using the following bit pattern operates the phone.

```
0000 0000
        ! !
        ! !—Bit zero, hook control.
        !       1 = on hook (power-up
                condition).
        !       0 = off hook.
        !
        !—      Bit one, dialer.
                1 = no pulse (power-up
                condition).
                0 = pulse line.
```

Listing 1 is the program I use to dial the phone. The program can be modified to add and display phone numbers. It operates the circuit by looping through a series of opening and closing the dialing relay to pulse the phone line.

Listing 2 is a modification of the first program. It dials a phone number at a predetermined time.

Program Listing 3 detects a ringing phone and answers it. When your computer answers the phone to communicate with another computer or terminal, it acts as a host, and you must have software that allows it to be used by another computer. Radio Shack has an RS-232 Communications Software package that performs adequately. I use LDOS as my main operating system and use the LDOS recommended host arrangement.

Program Listing 4 shows an LDOS JCL file that sets up the RS-232 driver and allows the computer to act as a host.

When running Program Listing 3, the computer answers the phone on the first ring. Several program lines check the time between responses and reset the phone after a period of no activity.

Listing 5 is a variation of Listing 3. I call it the ring-back method of answering the phone. As in most residences where only one phone line is present, this program will answer the phone according to a predetermined ringing sequence. If the phone rings only once and then rings again within 30 seconds, the computer answers the phone. The computer ignores the phone when it detects two rings in a row, as in typical phone calls.

This circuit works equally well with acoustic modems. If you have an acoustic modem, you can simply keep the handset in the modem, tape the hook buttons to the phone, and control the phone from the computer. My D-Cat hooks between the phone and handset, so I can leave the modem in the data mode, leave the handset in the cradle, and the phone still rings and operates normally. With other direct-connect modems, like the RS Modem I, you'll have to decide how to arrange the phone and the modem. The phone must be able to ring with the modem active, and the phone must be physically on hook, if only by the tape method.

To avoid legal problems, do not make connections directly to the telephone company's property when building this interface—make connections to your own phone. The altered phone should then be used through a phone coupler, available from your phone company. ■

Alan Moyer lives at 933 San Angelo Drive, Hamilton, OH 45013.

```
10 '**************************************************************
20 '**                   D I A L E R / B A S                   **
30 '**                                                         **
40 '** This program dials the phone after obtaining the        **
50 '** desired phone number from a menu.                       **
60 '**************************************************************
70 DEFINT A-Z: DIM NUM(20): XY = 7
80 FOR XX = 1 TO XY: READ PN$(XX): READ HN$(XX): NEXT XX
90 DATA "7718851","HBBS","8637681","XBBS","6712753","APPLE (PMS)
","1234567","TSO ","5790908","MICRONET","7918208","CINTUG","5314
588","Bam-Bam"
100 CLS: P0=254
110 OUT P0,3 'Put the phone on the hook
120 IF INP(PO%) AND 2 <> 2 THEN PRINT "Phone not put ON HOOK...r
etrying": LPRINT "Phone ON HOOK error at "TIME$:  GOTO 110
130 PRINT "Phone placed ON HOOK at " TIME$
140 PRINT: FOR XX = 1 TO XY: PRINT XX ") " HN$(XX): NEXT XX
150 PRINT " $ ) TO EXIT"
160 PRINT "** YOUR CHOICE...."
170 A$ = INKEY$: IF A$ = "" THEN 170
180 IF A$ = "$" THEN CMD"S"
190 XX = INSTR("1234567",A$): IF XX = 0 THEN 170
200 A$ = PN$(XX): PRINT: PRINT "CALLING: " HN$(XX) " @ " RIGHT$(
TIME$,8)
210 OUT P0,2  'Phone OFF HOOK
220 A = INP(P0) AND 2: IF A <> 0 THEN PRINT "Phone NOT OFF HOOK.
..retrying": LPRINT "Phone NOT OFF HOOK...ERROR @ " TIME$ "...re
trying": GOTO 210
230 FOR Z = 1 TO 100: NEXT Z  'Wait for dial tone
240 FOR X = 1 TO LEN(A$)
250 NUM(X) = ASC(MID$(A$,X,1)) - 48
260 IF NUM(X) = 0 THEN NUM(X) = 10
270 NEXT X
280 FOR X = 1 TO LEN(A$)
290 IF NUM(X) = 10 THEN PRINT 0; :ELSE PRINT NUM(X);
300 FOR Y = 1 TO NUM(X)
310 OUT P0,0  'Toggle line open
320 FOR Z = 1 TO 10: NEXT Z
330 OUT P0,2  'Toggle line on
340 FOR Z = 1 TO 10: NEXT Z
350 NEXT Y
360 FOR Z = 1 TO 100: NEXT Z 'Wait between numbers
370 NEXT X
380 PRINT: PRINT "Waiting for answer"
390 A = VAL(RIGHT$(TIME$,2)): IF A > 29 THEN A = A - 30 ELSE A =
 A + 30
400 IF INKEY$=" " THEN 430    'Abort if a key is pressed
410 C = INP(&HE8) AND 32: IF C = 0 THEN PRINT: PRINT "ON LINE...
Connection established..": END
420 B = VAL(RIGHT$(TIME$,2)): PRINT@896, A; B;: IF B <> A THEN 4
00 ELSE PRINT
430 OUT P0,3  'Put phone back onto hook
440 IF INP(P0) AND 2 <> 2 THEN PRINT "NOT able to put phone ON H
OOK...retrying": LPRINT "NOT able to put phone ON HOOK at " TIME
$ "....retrying": GOTO 430
450 PRINT: PRINT "Connection NOT ESTABLISHED....Press <ENTER> to
 continue...";: INPUT A$: RUN
```

Program Listing 1

Program Listing 2

```
10 '*********************************************************
20 '**                 C A L L A L A N / B A S             **
30 '**                                                     **
40 '** This program dials a number at a predetermined time. **
50 '** After thirty seconds, it puts the phone back on hook **
60 '*********************************************************
70 DIM NUM(20)
80 CLS
90 P0 = 254
100 XX = 0: OUT P0,3 'Put the phone on the hook
110 IF INP(P0) AND 2 <> 2 THEN XX = XX + 1: PRINT "Phone not put
 ON HOOK...retrying": LPRINT "Phone ON HOOK error at " TIME$ : I
F XX < 10 THEN GOTO 100 ELSE END
120 PRINT "Phone placed ON HOOK at " TIME$
130 IF RIGHT$(TIME$,8) = "07:30:00" THEN GOTO 140 ELSE 130
140 A$ = "xxxxxxx"    'Phone number goes here
150 PRINT "Calling ALAN at work at " TIME$
160 XX = 0: OUT P0,2  'Phone OFF HOOK
170 A = INP(P0) AND 2: IF A <> 0 THEN XX = XX + 1: PRINT "Phone
NOT OFF HOOK...retrying": LPRINT "Phone NOT OFF HOOK...ERROR @ "
 TIME$ "...retrying": IF XX < 10 THEN GOTO 170 ELSE END
180 FOR Z = 1 TO 100: NEXT Z  'Wait for dial tone
190 FOR X = 1 TO LEN(A$)
200 NUM(X) = ASC(MID$(A$,X,1)) - 48
210 IF NUM(X) = 0 THEN NUM(X) = 10
220 NEXT X
230 FOR X = 1 TO LEN(A$)
240 PRINT NUM(X);
250 FOR Y = 1 TO NUM(X)
260 OUT P0,0  'Toggle line open
270 FOR Z = 1 TO 10: NEXT Z
280 OUT P0,2  'Toggle line closed
290 FOR Z = 1 TO 10: NEXT Z
300 NEXT Y
310 FOR Z = 1 TO 100: NEXT Z 'Wait between numbers
320 NEXT X
330 PRINT: PRINT "Waiting for answer"
340 A = VAL(RIGHT$(TIME$,2)): IF A > 29 THEN A = A - 30 ELSE A =
 A + 30
350 B = VAL(RIGHT$(TIME$,2)): PRINT@256, A; B;: IF B <> A THEN 3
50 ELSE PRINT
360 XX = 0: OUT P0,3  'Put phone back onto hook
370 IF INP(P0) AND 2 <> 2 THEN XX = XX + 1 :PRINT "NOT able to p
ut phone ON HOOK...retrying": LPRINT "NOT able to put phone ON H
OOK at " TIME$ "....retrying": IF XX < 10 THEN GOTO 360 ELSE END

380 RUN "RINGBACK/BAS
```

Program Listing 3

```
10 '*******************************************************
20 '**                 A N S W E R / B A S               **
30 '**                                                   **
40 '** This supervisory routine will monitor the phone   **
50 '** line for a ringing phone.                         **
60 '*******************************************************
70 CMD"B","OFF"    'Disable the BREAK key in LDOS
80 DEFINT A-Z
90 TRUE = -1
100 FALSE = 0
110 P0 = 254
120 DEF FN XC$(A$) = CHR$((223) AND ASC(A$))  'Lower to upper co
nv.
130 CLS: PRINT "** Phone silent **"
140 C = INP(P0) AND 1
150 IF (VAL(MID$(TIME$,10,2)) < 8) AND (C = 1) THEN 140
160 IF (VAL(MID$(TIME$,10,2)) = 8) THEN PRINT "****** TIMED OUT
@ " TIME$ " ******": CMD"B","ON": CMD"S"  'Abort at 8 o'clock
170 XX = 0: OUT P0,2  'Take phone off the hook
180 D = INP(P0) AND 2: IF D <> 0 THEN XX = XX + 1: LPRINT "Phone
 NOT OFF HOOK...@ " TIME$ "...retrying": IF XX < 10 THEN GOTO 18
0 ELSE END
190 'Wait for a key to verify hook up
200 PRINT "Press any key...."
210 A = VAL(RIGHT$(TIME$,2)): IF A <= 29 THEN A = A + 30 ELSE A
= A - 30
220 A$ = INKEY$: IF A$ <> "" THEN 240
230 B = VAL(RIGHT$(TIME$,2)): IF B = A THEN RUN ELSE 220
240 PRINT "Connection established...": PRINT
250 PRINT "Welcome..."
260 PRINT "  Logged on at " RIGHT$(TIME$,8) " on " LEFT$(TIME$,8
)
270 T=0
280 PRINT: PRINT "ENTER your password"
290 INPUT A$: IF LEN(A$) = 0 THEN 330
300 FOR X = 1 TO LEN(A$)
310 MID$(A$,X,1) = FN XC$(MID$(A$,X,1))
320 NEXT X
330 T = T + 1: IF T = 5 THEN 380
340 IF A$ <> "HOWDY" THEN PRINT CHR$(7) "** Password NOT matched
 **": GOTO 340
350 PRINT "***>>> PASSWORD MATCHED <<<***"
360 PRINT "Welcome on board, the system is yours"
370 CMD"B","ON": CMD"S"
380 PRINT "**SORRY** but you blew the password entry 5 times in
a row"
390 PRINT "bye-bye": OUT P0,3: RUN
400 A = VAL(RIGHT$(TIME$,2)): IF A <= 29 THEN A = A + 30 ELSE A
= A - 30
410 CLS: PRINT "** Waiting 30 seconds to purge this call **"
420 B = VAL(RIGHT$(TIME$,2)): IF B <> A THEN 420 ELSE 130
```

```
.********** HOST/JCL **********
.Install the RS-232-C driver
SET *CL RS232R
.Set up the I/O links for remote operation
LINK *KI *CL
LINK *DO *CL
```

Program Listing 4

Program Listing 5

```
10 '*************************************************************
20 '**                    R I N G   B A C K                    **
30 '**                                                          **
40 '** This supervisory routine will monitor the phone          **
50 '** line for a ringing phone.  If a second ring is           **
60 '** detected, the system is reset.  If there is no           **
70 '** second ring, the computer will answer the phone          **
80 '** if the bell rings again within thirty seconds.           **
90 '** Otherwise the system is again reset.                     **
100 '** Version 1.0 by Alan R. Moyer 09/10/81                   **
110 '*************************************************************
120 CMD"B","OFF"
  'Disable BREAK in LDOS
130 DEFINT A-Z
140 TRUE = -1
150 FALSE = 0
160 P0 = 254
170 DEF FN XC$(A$) = CHR$((223) AND ASC(A$))  'Lower to Upper co
nv
180 CLS: PRINT "** Phone silent **"
190 C = INP(P0) AND 1: IF C = 1 THEN 190
200 PRINT "1st ring detected"
210 A = VAL(RIGHT$(TIME$,2))     'Start countdown
220 PRINT "Waiting between the rings"
230 IF A <= 55 THEN A = A + 4 ELSE A = A - 56  'We'll wait 4 sec
onds between the rings
240 B = VAL(RIGHT$(TIME$,2)): PRINT B: PRINT CHR$(27);: IF B <>
A THEN 240
250 A = VAL(RIGHT$(TIME$,2)): IF A <= 55 THEN A = A + 4 ELSE A =
 A - 56  'We'll wait 4 seconds for the second ring
260 PRINT "Waiting to see if there is a second ring"
270 B = VAL(RIGHT$(TIME$,2)): PRINT B: PRINT CHR$(27);
280 C = INP(P0) AND 1: IF C = 0 THEN PRINT "2nd ring detected, i
gnoring this call": GOTO 570  'Go wait, this is not our call
290 IF B = A THEN PRINT "No second ring, next call will be answe
red" ELSE 270
300 A = VAL(RIGHT$(TIME$,2)):FOR Z=0 TO 720: NEXT   'We'll wait
60 seconds for the second call
310 C = INP(P0) AND 1: IF C = 0 THEN PRINT "2nd call recieved...
answering...": GOTO 340
320 B = VAL(RIGHT$(TIME$,2)): PRINT B: PRINT CHR$(27);: IF B <>
A THEN 310
330 PRINT "Timed out, no 2nd call recieved....resetting...": GOT
O 180
340 XX = 0
350 OUT PO,2
360 PRINT "Press any key..."
370 A = VAL(RIGHT$(TIME$,2)): IF A <= 29 THEN A = A + 30 ELSE A
= A - 30
380 A$ = INKEY$: IF A$ <> "" THEN 400
390 B = VAL(RIGHT$(TIME$,2)): IF B = A THEN RUN ELSE 380
400 PRINT "Connection established...": PRINT
410 PRINT "Welcome ..."
420 PRINT "  Logged on at " RIGHT$(TIME$,8) " on " LEFT$(TIME$,8
)
430 T = 0
440 PRINT: PRINT "ENTER your password"
450 INPUT A$: IF LEN(A$) = 0 THEN 490
460 FOR X = 1 TO LEN(A$)
470 MID$(A$,X,1) = FN XC$(MID$(A$,X,1))
480 NEXT X
490 T = T + 1: IF T = 5 THEN 550
500 IF A$ <> "HOWDY" THEN PRINT CHR$(7)"** Password NOT matched
**": GOTO 440
510 PRINT "***>>> PASSWORD MATCHED <<<***"
520 CMD"B","ON"  'Turn BREAK back on
530 PRINT "Welcome on board, the system is yours"
540 END
550 PRINT "**SORRY** but you blew the password entry 5 times in
a row"
560 PRINT "bye-bye": OUT P0,3: RUN
570 A = VAL(RIGHT$(TIME$,2)): IF A <= 29 THEN A = A + 30 ELSE A
= A - 30
580 PRINT "** Waiting 30 seconds to purge this call **"
590 B = VAL(RIGHT$(TIME$,2)): IF B <> A THEN 590 ELSE 180
```


✓See List of Advertisers on Page 563

120 DAY WARRANTY

"NO YOU'RE NOT SEEING THINGS, THAT'S FOUR INTERNAL DRIVES IN A MODEL III, WITH UP TO 4 MEG. DISK STORAGE AVAILABLE NOW, ONLY FROM DISPLAYED VIDEO!!"

NEW LOW PRICES

DISPLAYED VIDEO IS DRIVING DOWN PRICES ON DISKS!

NEW LOW PRICES

DISPLAYED VIDEO is now offering TRS-80* MODEL III with TANDON disk drives, one of the most reliable disk drive systems on the market, for **INCREDIBLY** low prices.

MODEL III with 48K, dual 40 track double density disk drives, complete systems with TRSDOS and ONE BOX OF DISKETTES

PLUG IT IN AND GO...... **$1,749.00**

MODEL III with 48K, dual 40/40 track double density disk drives, complete systems with NEWDOS 80 and ONE BOX OF DISKETTES

PLUG IT IN AND GO...... **$2,149.00**

MODEL III with 48K, dual 80 track double density disk drives, complete systems with NEWDOS 80 and ONE BOX OF DISKETTES

PLUG IT IN AND GO...... **$2,149.00**

MODEL III with 48K, dual 80/80 track double density disk drives, complete systems with NEWDOS 80 and ONE BOX OF DISKETTES

PLUG IT IN AND GO...... **$2,449.00**

MODEL III with 48K, four 40 track double density internal disk drives, complete systems with TRSDOS and ONE BOX OF DISKETTES

PLUG IT IN AND GO.... **$2,399.00**

MODEL III with 48K, four internal drives of any configuration available to achieve up to 4 meg of disk storage .... **CALL**

MODEL III Internal Drive Kit: Includes controller board, dual drive mounting bracket, dual power supply, all hardware cables and connectors (gold plated) **$329.00**

MODEL 16 w/1 drive....**$4,450.00**.........w/2 drives.............**$4,850.00**

IMMEDIATE DELIVERY
120 Days Parts and Labor Warranty
DEALER INQUIRIES INVITED
Free Shipping in the U.S. 48 Contiguous States

Visit Our Two RETAIL LOCATIONS:
886 Ecorse Rd., Ypsilanti, MI 48197
(313) 426-5086/(313) 482-4424
111 Marshall St., Litchfield, MI 49252
(517) 542-3280

DISPLAYED VIDEO

Radio Shack Authorized Dealership at:
111 Marshall Street, Litchfield, Michigan 49252 ***#P142***

To Order: Call (313) 426-5086 or (313) 482-4424 or (517) 542-3280 or write DISPLAYED VIDEO, 886 Ecorse Road, Ypsilanti, Michigan 48197

*TRS-80 is a trademark of the Tandy Corporation

Prices subject to change without notice

DISPLAYED VIDEO IS DRIVING DOWN PRICES ON DISKS!

"PRICE" "CUT"

DV'S COLOR COMPUTER FIRST DRIVE
$425.00

80 TRACK TEAC DISK DRIVE W/CASE & POWER SUPPLY
$275.00

40 TRACK TEAC DISK DRIVE W/CASE & POWER SUPPLY
$239.00

TEAC'S NEW 40 TRACK SLIMLINE W/CASE & POWER SUPPLY
$269.00

"PRICE" "CUT"

ONE YEAR WARRANTY ON TEACS

TWO DRIVE CABLE $23.99

With Gold Plated Connectors

NEWDOS 80 $115.00 MODEL I or III

Bare Drives & Cases Available Upon Request

DV'S 5 MEG. HARD DISK
$1949.00

ATHANA DISKETTS
$26.95

100 Percent Certified W/Own Hub Rings

EPSON PRICES

We're not going to make you call for prices.
Here they are in black and white!!

MX 80 (W/GRAPHTRAX) PLUS $459.00
MX80FT (W/GRAPHTRAX) PLUS $545.00
MX82FT (W/GRAPHTRAX) PLUS $555.00
MX100 (W/GRAPHTRAX) PLUS $709.00
PRINTER CABLE $23.99

Stop In Our Two Retail Locations:
886 Ecorse Rd., Ypsilanti, MI 48197
(313) 426-5086/(313) 482-4424
or
111 Marshall Street,
Litchfield, MI 49252
(517) 542-3280

We Carry A Full Line of OKIDATA Printers

DISPLAYED VIDEO

FREE SHIPPING!!

IMMEDIATE DELIVERY
180 Days Parts and Labor Warranty
DEALER INQUIRIES INVITED
Free Shipping in the U.S.
48 Contiguous States

Radio Shack Authorized Dealership at:
111 Marshall Street, Litchfield, Michigan 49252 *#P142*

To Order: Call (313) 426-5086 or (313) 482-4424 or (517) 542-3280 or write DISPLAYED VIDEO, 886 Ecorse Road, Ypsilanti, Michigan 48197

*TRS-80 is a trademark of the Tandy Corporation

Prices subject to change without notice

✓62

Fly Like an Eagle

by Ian Cohn

Use this simulation of a sailplane race on your Model I to learn the basics of the sport. It'll show you the thrill of being in the air.

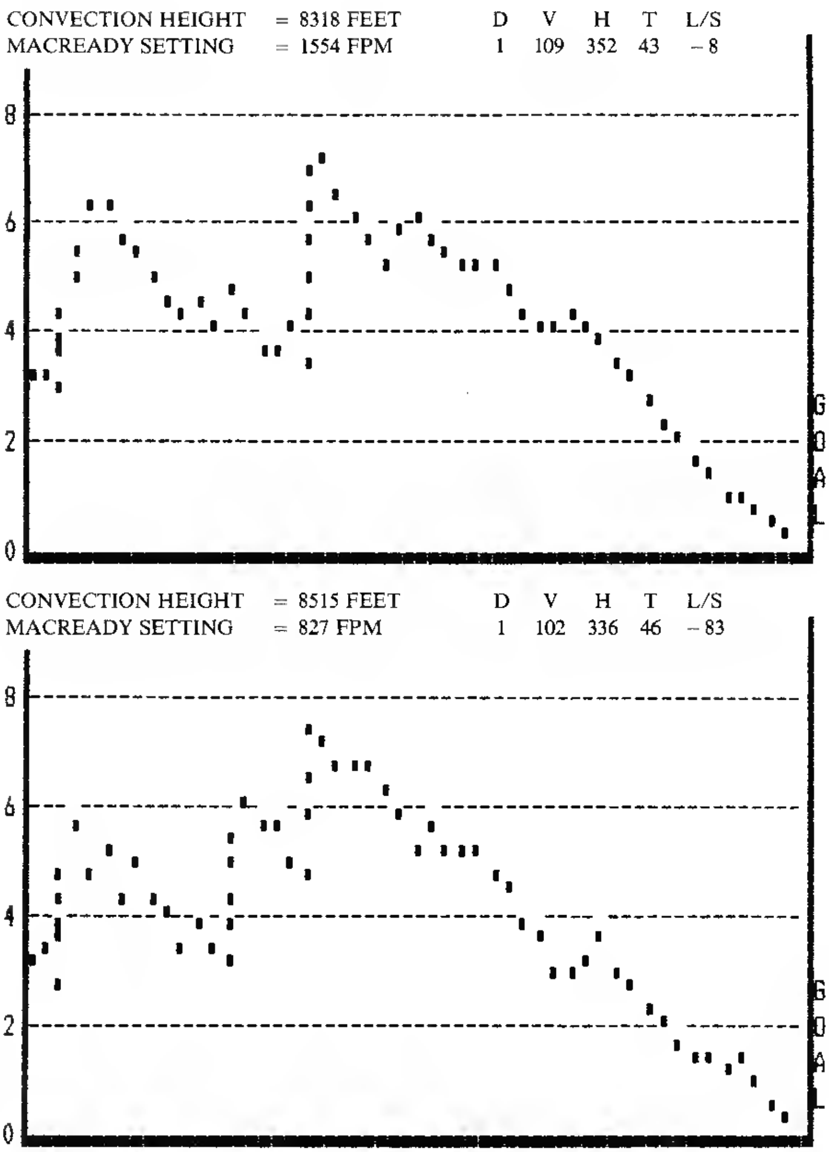


Figure 1

This program is a simulation of a sailplane race, but nevertheless includes most of the elements of the real thing.

Sailplanes are fixed wing, motorless aircraft propelled by the earth's gravitational attraction. They are distinguished from gliders by their ability to use upcurrents in the atmosphere to maintain height and travel long distances. Modern sailplanes are used in many countries for races over 200 km and 500 km distances. Often, speeds of 140 km per hour or higher are achieved.

Sailplane races start with the sailplane flying through a start gate. The pilot flies toward the goal looking for an upcurrent. A normal type of upcurrent is a column of rising air called a thermal. Thermals rise from ground heated by the sun at a different rate than the ground nearby.

When the pilot finds a sufficiently strong thermal, he circles to gain altitude. Eventually the climb rate reduces and the pilot straightens out and flies on. This cycle repeats until the pilot gets close enough to the goal to glide to the finish line without further circling.

A sailplane pilot relies to a great extent on his flight instruments (shown in Table 1). In addition to these, an instrument called an audio variometer can be used. The audio variometer gives a series of beeps, with the frequency indicating the lift's strength.

In the simulator program, lines 890–1330 contain the sailplane flight

The Key Box

Model I
16K RAM
Cassette Basic

manual. Lines 1340–1500 contain a preflight check; this check establishes the sailplane configuration, the race task, and the start-gate speed and height. The pilot aims to go through the start gate close to the maximum allowed height (3,280 feet) and the sailplane's maximum speed (130 knots).

Finally, a random number establishes the atmosphere for the day and the computer notes the pilot's name. If A is the name, the autopilot (George), flies the task.

After completing the preflight check, the computer draws the sky and the instrument panel (lines 1510–1570). The sky is a graph of height versus distance with height reference lines at 2,000-foot intervals up to 8,000 feet.

The computer launches the sailplane and establishes the strength of the up or down current for the next mile in the weather-generator routine (lines 160–200). The weather generator reduces the strength of the air lift/sink near the ground and near the convection height. The weather generator also ensures that the air lift and sink are balanced. The average air lift is twice as strong as the air sink, but sink occurs twice as often as lift. If the lift is above the threshold value, the audio variometer gives a beep.

The sailplane position is set on the screen, and the instrument panel updates itself (lines 210–240). Now you find out if you've crossed the finish line (D = 0) or landed before reaching the goal (H<0), and whether George is flying (lines 250–280).

The pilot now inputs his commands for the next segment of the flight. His options include:

- Fly straight at his selected speed. You can achieve this by entering a speed as a three-item number between the stall speed (45 knots with water ballast or 40 knots dry) and the maximum permitted speed (130 knots)—for example, 067 for a speed of 67 knots.
- Circle for one minute. Select this by entering O or C. The sailplane speed reduces to 5 knots above the stall speed when circling.
- Dump water ballast and circle, selected by entering D. The sailplane circling sink rate is 140 feet per minute when dry or 187 feet per minute with ballast. Water ballast is dumped only if the pilot is low and circling in weak lift.
- Select a pseudo-optimum (MacReady) speed to fly straight for one mile by entering a negative three-item number—for example, -05. This represents the pilot's estimate of his actual climb rate in the next thermal (climb rate in feet per minute divided by 100. In this case, -05 means an estimate of 500 feet per minute for the next climb).
- The keyboard data-entry routine (lines 290–430) uses the INKEY$ function within For. . . Next loops. This allows the audio variometer to continue beeping while the computer waits for keyboard input. While the computer beeps, hold the appropriate key down

Altimeter—	Gives height in feet (above ground in this case).
Air Speed— Indicator	Shows current speed in knots.
Clock—	Shows time in minutes since start.
Air lift—	Shows air vertical speed in feet per minute.
Distance— Indicator	Gives distance to goal in nautical miles.
MacReady— Setting	Shows current setting of optimum speed-to-fly indicator.
Convection—	Gives height at which thermal strength reduces to zero.

Table 1. Flight Instruments

Program Listing. Sailplane Simulator

```
100 GOTO 840 '
                    SAILPLANE SIMULATOR
                            by
                       IAN   COHN
110 '
     **** keyboard entry routine ****

120 PRINT@Z,C$;:FOR X1=0 TO X2:Z$=INKEY$:IF Z$<>"" THEN X1=X2
    :NEXT ELSE NEXT:IF C$=A$ THEN C$=F$ ELSE C$=A$
130 RETURN
140 '
     ****  sound generator  ****

150 Z=VARPTR(V$):Y=PEEK(Z+1)+PEEK(Z+2)*X:Z=INT(Y/X):POKE16527,Z
   :POKE16526,Y-Z*X:F=9+4*L:FC=INT(4E4/F):DC=F/29:POKEY+6,FC/X
   :POKEY+5,FC-INT(FC/X)*X:POKEY+3,DC/X:POKEY+2,DC-INT(DC/X)*X
   :Z=USR(0):RETURN
160 '
    **** weather generator ****

170 N=(N+PI+.2/(D+1))[4:N=N-INT(N):LB=(K+I/4)*(N-2/3)
   :IF LB<0 THEN LB=LB/4
180 L=LB:IF H<K2 THEN L=H*L/K2
190 IF H>I-K2 THEN L=L*(I-H)/K2
200 IF L>LT GOSUB 150
210 '
    **** locate sailplane on screen ****

220 IF H>0 THEN SET(4+120*(1-D/DS),10+36*(1-H/8000))
230 '
    **** update instrument panel ****

240 PRINT@99,USING B$;D;V;H;T;L;
250 '
    **** have we crashed or finished? ****

260 IF H<=0 GOTO 560 ELSE IF D=0 GOTO 570
270 '
    **** is "GEORGE" flying? ****

280 IF M$="A" GOTO 760
290 '
    **** look for and display keyboard data entry ****

300 X2=9:PRINT@122,"?   ";:C$=A$:IF L<LT THEN X2=44
310 Z=123:GOSUB 120:I$=Z$:IF I$<>"" GOTO 330
320 IF L<LT GOTO 310 ELSE GOSUB 150:GOTO 310
330 PRINT@Z,I$;:IF I$="D" OR I$="d" GOTO 620
340 IF I$="O" OR I$="o" OR I$="C" OR I$="c" THEN E=0:GOTO 640
360 IF I$=E$ THEN V=E:GOTO 450
370 Z=124:GOSUB 120:J$=Z$:IF J$<>"" GOTO 390
380 IF L<LT GOTO 370 ELSE GOSUB 150:GOTO 370
390 IF J$="E" OR J$="e" THEN 300 ELSE PRINT@Z,J$;
400 Z=125:GOSUB 120:K$=Z$:IF K$<>"" GOTO 420
410 IF L<LT GOTO 400 ELSE GOSUB 150:GOTO 400
420 IF K$="E" OR K$="e" PRINT@Z," ";:GOTO 370
430 PRINT@Z,K$;:V=VAL(I$+J$+K$)
```

Listing continues

WE STOCK DISK DRIVES!

TEAC

BARE		COMPLETE WITH POWER SUPPLY & CABINET	
40 Track, Single	$209	40 Track, Single	$239
40 Track, Double	$289	40 Track, Double	$340
80 Track, Single	$289	80 Track, Single	$340
80 Track, Double	$349	80 Track, Double	$399

TANDON

BARE		COMPLETE WITH POWER SUPPLY & CABINET	
40 Track, Single	$209	40 Track, Single	$239
40 Track, Double	$269	40 Track, Double	$319
80 Track, Single	$269	80 Track, Single	$319
80 Track, Double	$319	80 Track, Double	$369

We sell two brands of disk drives for the TRS-80: the TEAC and the Tandon. The TEAC disk drive has a 1 year warranty, the Tandon has a 90 day warranty. The TEAC uses a lead screw actuator, the Tandon uses a split band type actuator. The track to track access rate for the Tandon is 5 milliseconds, as opposed to 25 milliseconds for the TEAC. The TRS-80 Model III requires a faster drive speed than the Model I, therefore the Tandon works better with the Model III. With the slower drive speed requirements of the Model I, the TEAC is a more reliable drive. We have both 40 track and 80 track drives with either single or dual head. All drives are capable of double density. These drives are available either bare or complete with power supply and cabinet. A cable is required to hook up the drives. We have both two-drive and four-drive cables. All drives come with complete instructions for hooking up a system. TEACs and Tandons can be intermixed with other drives on the same system.

DISK DRIVES for the TRS-80 MODEL III

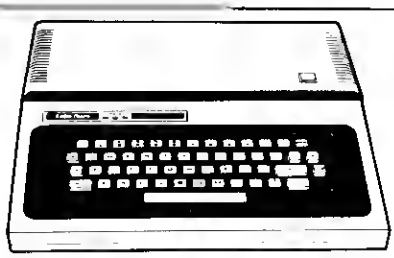
DRIVE 0		DRIVE 1	
Single Side, 40 Track	$595	Single Side, 40 Track	$219
Double Side, 40 Track	$729	Double Side, 80 Track	$279
Single Side, 80 Track	$729	Single Side, 40 Track	$279
Double Side, 80 Track	$819	Double Side, 80 Track	$329

The disk drives we sell for the Model III are the Tandon drives. Drive 0 includes the controller board, power supply, cables and all mounting hardware. Complete instructions are included for installation. It takes 30 minutes to an hour to install disk drives in a Model III. No soldering is required. TRSDOS operating system is not included in this low price.

IBM PERSONAL COMPUTER DISK DRIVES

SINGLE SIDE, 40 TRACK	$209
DOUBLE SIDE, 40 TRACK	$269

These drives are fully compatible with the IBM Personal Computer. They can be used as the first or second drive and come with our exclusive 90 day warranty. DOS and Drive Adapters required. Our disk drives adhere to the high quality standards that IBM demands.



16K LEVEL I

NOW ONLY $279

COLOR COMPUTERS

ALSO AVAILABLE:

16K OR 32K EXTENDED BASIC

COLOR COMPUTER DISK DRIVES

CALL NOW FOR SPECIAL PRICES

THESE ARE OUR CASH DISCOUNTED PRICES. C.O.D. AND CHARGE ORDERS ARE 3% HIGHER. ALL COMPUTERS ARE SHIPPED FREIGHT COLLECT. PRICES, SPECIFICATIONS AND AVAILABILITY ARE SUBJECT TO CHANGE WITHOUT NOTICE. IBM & IBM PERSONAL COMPUTERS ARE TRADEMARKS OF INTERNATIONAL BUSINESS MACHINES, INC. TRS-80 AND RADIO SHACK ARE TRADEMARKS OF TANDY CORPORATION.

918/825-4844

AMERICAN Small Business COMPUTERS

118 SO. MILL ST
PRYOR, OK 74361

Listing continued

```
440 '
     **** circling option ****
450 E=V:IF V=0 GOTO 640
460 '
     **** Beep if in lift ****

470 IFL>LTGOSUB150
480 '
     **** McReady optimum speed ****

490 IF V<0 GOTO 670 ELSE PRINT@84,"    ";
500 '
     **** speed limiter ****

510 IF V<VS THEN V=VS ELSE IF V>130 THEN V=130
520 '
     **** increment flight parameters and BEEP if in lift ****

530 T=T+60/V:S=A1*V*V*V+B/V:IF L>LT GOSUB 150
540 H=H+60*(L-S)/V+(U*U-V*V)/23:U=V:D=D-1:GOTO 170
550 '
     **** flight completion data ****

560 PRINT@32,"LANDED AT";D;" MILES  TO  GO.":V(NA)=NA/9999:GOTO
590
570 Z$="ELAPSED TIME =####.# MIN.      AVERAGE SPEED ="
580 V(NA)=111.2*DS/T+NA/999:PRINT@0,USINGZ$+"###.# KPH.       ";T
;V(NA)
   :PRINTUSING"FINISH HEIGHT =##### FT.       ";H;
590 D(NA)=DS-D:T(NA)=T:H(NA)=H:GOSUB 1900:PRINT"COMMAND?";
600 CC$=INKEY$:IF CC$="" THEN 600 ELSE GOTO 1720
610 '
     **** dump water ballast ****

620 W=6:VS=40:E=0:A1=.34*C/W:B=9510*W/A
630 '
     **** circle and BEEP if in lift ****

640 PRINT@84,"    ";:V=VS+5:H=H+L-140*W/6+(U*U-V*V)/23:IFL>LTGOS
UB150
650 U=V:T=T+1:IF H<0 THEN GOTO 560 ELSE GOTO 180
660 '
     **** solve McReady equation ****

670 IF M$<>"A" THEN V=100*V
680 IF V<-K2 THEN V=-K2
690 PRINT@84,USING"####";-V;:Z=-V-L:C9=Z/2/A1:Z9=SQR(C9[4+4*(4*B
/3/A1)[3)
700 Z7=(C9*C9+Z9)/2:Z8=-(C9*C9-Z9)/2:P2=Z7[(1/3)-Z8[(1/3)
   :P=SQR(P2):IF Z<0 THEN P=-P
710 IF P>0 THEN V=(P+SQR(2*C9/P-P2))/2 ELSE V=VS
720 V=-V*(V>VS)-VS*(V<=VS)
730 IF M$="A" THEN V=(V/2+U/D)/(1/2+1/D)
740 GOTO 510
750 '
     **** autopilot ****

760 J=6080*(D+2)/(H+L+U*U/30):L1=1/C/(3+I/9999):IF J>L1 GOTO 800

770 V=(K-500/C/J)*(.4+.6*(1-1/D/D))
780 IF L>0 THEN V=V-L*(20/J+1/D)
790 GOTO 670
800 Z=20*W+H/10:IF L<230 AND L>130 AND H<K GOTO620
810 IF L>Z GOTO 640
820 V=0-(1+H/10+2*L/D/D/D):IF V>-1 OR H<K THEN V=-1
830 GOTO 670
840 CLS:PRINT"
       ****************************************
       *          SAILPLANE  SIMULATOR        *
       *                 by                   *
       *              Ian  Cohn               *
       ****************************************"
850 '
     **** initialise constants and poke sound routine *****

860 CLEAR999:DEFINTD,I,X:DIM NA$(99),D(99),V(99),T(99),H(99)
   :X=256:X2=999:A$=CHR$(143):E$=" ":F$=CHR$(95):V$=""
   :PRINT:FOR I=0 TO 32:READ Z:V$=V$+CHR$(Z):NEXT
870 INPUT"Do you want to read the Flight Manual";Z$:IF LEFT$(Z$,
1)<>"Y"
   GOTO 1330 ELSE GOSUB 890:GOTO 1330
880 '
     **** flight manual - first page ****

890 CLS:PRINT"FLIGHT MANUAL";TAB(58)"Page 1":PRINT
900 PRINT"This program simulates cross-country flight in a Sailp
lane."
```

Listing continues

until it registers and appears on the instrument panel. This is not a problem if the audio variometer is switched off.

The main flight-progress calculations are carried out in lines 440–650. The cruise flight routine at lines 530 and 540 increments the time according to the selected speed, subtracts one mile from the distance to go, and calculates the height at the end of the one-mile segment based on the commencing height, the air lift/sink, the sailplane sink rate at the selected speed, and the speed change.

Line 620 dumps the water ballast, and lines 640 and 650 calculate the circling flight parameters. Set the speed to five knots above the stall speed, adjust the height, and increment the time by one minute.

If the sailplane reaches the finish line or lands prematurely, the flight's final results are given (lines 560–600). A set of commands is then available so you can look at the scoreboard, call the next pilot, and so on. The commands routine is at lines 1620–1800. The command H displays all the available commands.

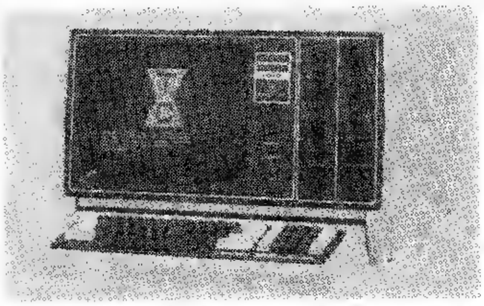
Lines 1820–1880 generate the scoreboard. The results are arranged from best to worst by a simple bubble-sort routine (lines 1900–1960). This sort is slow when many pilots are racing but I leave to modifiers to implement a faster sort.

Another section of the simulator is the optimum-speed generator (originated by Dr. Paul MacReady). The optimum speed needed to fly between thermals depends on the climb rate the pilot achieves in the next thermal. Since the pilot never knows his exact climb rate for the next ascent, he has to estimate. In general, the higher the anticipated climb rate, the faster you fly between climbs. Also, the heavier the sink, the faster you fly. The routine uses both these elements to calculate an estimate of the optimum speed. If you choose this option, the sailplane immediately flies the next mile at the calculated speed.

The autopilot (lines 750–830) uses the optimum-speed routine exclusively to determine the flying speed. The philosophy behind the autopilot is that the higher the sailplane goes, the faster it flies, and the higher it is, the stronger the lift should be before used for climbing.

When the autopilot nears the finish line, it varies the MacReady lift setting to maintain constant glide slope to the finish line. The autopilot algorithm is not optimized, and although it pro-

TRS-80 MODEL 16



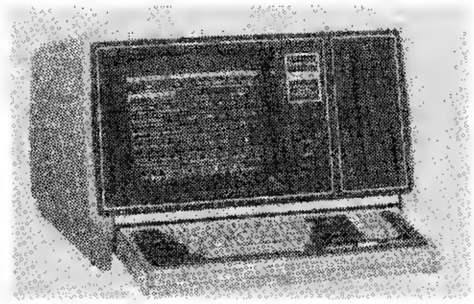
MODEL 16-1 DRIVE $4499

MODEL 16-2 DRIVE $4995

DT-1 VIDEO DATA TERMINAL $620

For use with Model 16 or Mainframe Computer

TRS-80 MODEL II



Now Only $2895

NEW! HARD DISK DRIVES

HARD DISK SYSTEM FOR THE MODEL III

6 MEG $1795

12 MEG $1995

19 MEG $2195

COMPLETE, READY TO RUN. NOW AVAILABLE!!

LDOS REQUIRED — $129

*unformatted capacity

CAMEO SUBSYSTEMS FOR THE MODEL II

DRIVE MOUNT ONE $4995—
- Western Dynex D6222 Drive (5 mb Fixed/5 mb Removable) Controller, Adaptor and Cables

1010 SUBSYSTEM $4995—
- Data Peripheral Lynx Drive, (10 mb Removable Winchester) Controller, Adaptor and Cables

1040 SUBSYSTEM $5995—
- Data Peripheral Puma Drive (40 mb Fixed Winchester) Controller, Adaptor and Cables

1020 SUBSYSTEM $7995—
- 2 Data Peripheral Lynx Drives (20 mb Removable Winchester) Controller, Adaptor and Cables

1050 SUBSYSTEM $8995—
- Data Peripheral Lynx and Puma Drive (40 mb Fixed/10 mb Removable) Controller, Adaptor and Cables

—EXTRA DRIVES—

WESTERN DYNEX D622 $4495

DATA PERIPHERAL LYNX $3995

DATA PERIPHERAL PUMA $4995

MULTIPLEXORS

MX200-2 PORT MULTIPLEXOR $495

(Cables and Adaptors Not Included)

MX400-4 PORT MULTIPLEXOR $750

(Cables and Adaptors Included)

THESE ARE OUR CASH DISCOUNTED PRICES. C.O.D. AND CHARGE ORDERS ARE 3% HIGHER. ALL COMPUTERS ARE SHIPPED FREIGHT COLLECT. PRICES, SPECIFICATIONS AND AVAILABILITY ARE SUBJECT TO CHANGE WITHOUT NOTICE. IBM & IBM PERSONAL COMPUTERS ARE TRADEMARKS OF INTERNATIONAL BUSINESS MACHINES, INC. TRS-80 AND RADIO SHACK ARE TRADEMARKS OF TANDY CORPORATION.

918/825-4844

AMERICAN Small Business COMPUTERS

118 SO. MILL ST
PRYOR, OK 74361

duces a good average speed, it can be beaten by a human pilot.

The last routine is the sound generator at line 150. This is a modification of Bill Barden's tone subroutine (*80 Micro*, April 1980). I've made the sound generator relocatable and imbedded it in a string so that no memory needs to be reserved.

The sound output goes to the number one cassette port 255, and can be heard by connecting the microphone cable to a small amplifier. ■

Ian Cohn can be reached at 1 Manor St., Brighton, Victoria, Australia.

Glossary of Terms

Sailplane Performance: The universal measure of sailplane performance is the maximum glide ratio. This is the maximum number of distance units traveled for each unit of height lost in still air. For example, a sailplane having a glide ratio of 40 would be able to travel 40 miles from a height of one mile (5,280 feet) in still air.

Water Ballast: Water ballast improves the sailplane's performance at high forward speed. You obtain a lower sink rate at high speed at the expense of a higher sink rate at low speed and a higher stall speed. The net result is that it reduces the climb rate in thermal lift, and also reduces the time spent in cruising between thermals.

Generally, ballast is carried only when strong thermal lift is expected. Under these conditions, the average speed can be maximized by carrying ballast. If weak lift is all that's available, dump the water ballast by opening the taps.

Sailplane Class: Sailplanes are classified according to their performance category. Standard class sailplanes are allowed a maximum wing span of 15 meters and are not allowed to use wing flaps. The 15-meter class sailplanes have only one limitation, which is a wing span of no more than 15 meters. Open class sailplanes have no limitations.

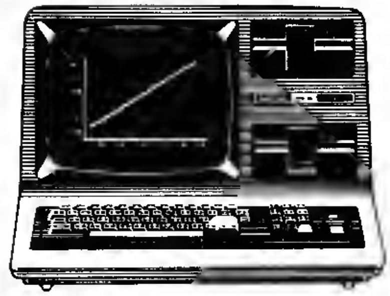
Standard class sailplanes have a maximum glide ratio of between 38 and 40, 15-meter class between 39 and 42, and open class sailplanes between 45 and 50.

Listing continued

```
910 PRINT"The aim of the pilot is to fly to a goal, without"
920 PRINT"landing enroute, preferably in the shortest possible t
ime,"
930 PRINT"using the upcurrents (lift) and downcurrents (sink) in
 the"
940 PRINT"simulated atmosphere to the best advantage. The pilot
may race"
950 PRINT"against other pilots or against the inbuilt autopilot
'GEORGE'.":PRINT
960 PRINT"In this program only the leftmost letter is required f
or any"
970 PRINT"alpha response, and defaults are provided.":PRINT
    :PRINT:GOSUB 1590:IF G RETURN
980 '
    **** flight manual - second page ****

990 CLS:PRINT"FLIGHT MANUAL";TAB(58)"Page 2"
    :PRINT:PRINT"The initial data required is:-":PRINT
1000 PRINT"Audio                Y for sound, N for silence"
1010 PRINT"Sailplane Class      'Open', '15m.', or 'Std.' (Default-
Open)"
1020 PRINT"Water Ballast        'Yes' or 'No' (Default-Yes)"
1030 PRINT"Start Speed          Any speed between the minimum (40 k
t.-dry"
1040 PRINT"                     or 45 kt.-wet) and 130 kt. (Default
-120 kt.)"
1050 PRINT"Start Height         Any height (Default-3280 ft.)"
1060 PRINT"Distance to go       Any integer distance (Default-50 n.
m.)"
1070 PRINT"Pilot Name           Any name of less than 12 characters

1080 PRINT"                     The name 'A' operates the autopilot
 'GEORGE'"
1090 GOSUB 1590:IF G RETURN
1100 '
1110 CLS:PRINT"FLIGHT MANUAL";TAB(58)"Page 3";
1120 PRINT"The display now shows the DISTANCE TO GO (D), SPEED (
V), HEIGHT"
1130 PRINT"(H), elapsed TIME (T), and the AIR VERTICAL SPEED (L/
S) for the"
1140 PRINT"next mile. The pilot now has the following options:-"

1150 PRINT" -to cruise for the next mile, input any speed betwee
n the"
1160 PRINT"  the minimum speed and 130 kt. eg; 0 7 5 (3 digits)
for 75 kt."
1170 PRINT" -to circle, preferably in air lift greater than the
Sailplane"
1180 PRINT"  sink rate, input 'O' or 'C'. The Sailplane circling
 sink rate"
1190 PRINT"  is 140 FPM (dry) or 187 FPM (wet)."
1200 PRINT" -to dump water ballast and circle, input 'D'."
1210 PRINT" -to fly at the estimated best (McReady) speed, input
 a"
1220 PRINT"  negative estimate of the actual climb rate in the n
ext"
1230 PRINT"  thermal in FPM/100. eg; - 0 3 (3 items) for 300 FPM
"
1240 PRINT" -to repeat the last input, press the Space Bar."
    :GOSUB 1590:IF G RETURN
1250 '
    **** flight manual - fourth page ****

1260 CLS:PRINT"FLIGHT MANUAL";TAB(58)"Page 4
  When entering a cruising speed or McReady setting, 'E' may be
used to ERASE the first 2 items if a mistake has been made.

  The display is repeated until the flight is completed. This"
1270 PRINT"occurs when the Sailplane reaches the goal or when it
 lands"
1280 PRINT"prematurely. A set of optional COMMANDs is available
to display"
1290 PRINT"the scoreboard and continue the next flight."
1300 PRINT"The COMMAND 'H' will list the available commands."
1310 PRINT:PRINT"Press any key to continue";
1320 Z$=INKEY$:IFZ$="" GOTO 1320 ELSE CC$="H":RETURN
1330 PRINT:PRINT:PRINT"Now you can start!":PRINT
1340 '
    **** pre-flight check ****

1350 PRINT:INPUT"Audio Variometer (Y OR N)";Z$
1360 LT=140-9999*(Z$="N"):L=888:ZZ$=STRING$(60,"-"):GOSUB150
1370 INPUT"Sailplane Class - Open, 15m., or Standard";C$:IF
    C$="" THEN C$="O" ELSE C$=LEFT$(C$,1)
1380 Y=38:K=1000:K2=K*2:GOSUB150:W=8
```

Listing continues

TRS-80 ★ MODEL III



BRAND NEW, 180 DAY WARRANTY

★ 16K $825.00 ★

★ 48K $849.00 ★

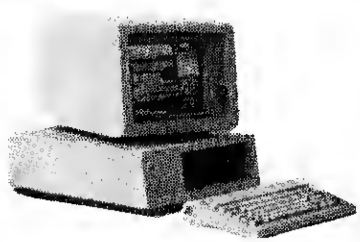
★ CASSETTE RECORDER $55.00 ★

★ 48K, 1 DR $1499 ★

★ 48K, 2 DR $1749 ★

★ RS-232 (INSTALLED) $92 ★

DUAL SIDED DISK DRIVES ALSO AVAILABLE



IBM P. C.

THESE ARE BRAND NEW IBM PERSONAL COMPUTERS

64K WITH 2 INTERNAL DUAL SIDE DISK DRIVES

DISCOUNT PRICE!!! ONLY $2699.95

THESE ARE OUR CASH DISCOUNTED PRICES. C.O.D. AND CHARGE ORDERS ARE 3% HIGHER. ALL COMPUTERS ARE SHIPPED FREIGHT COLLECT. PRICES, SPECIFICATIONS AND AVAILABILITY ARE SUBJECT TO CHANGE WITHOUT NOTICE. IBM & IBM PERSONAL COMPUTERS ARE TRADEMARKS OF INTERNATIONAL BUSINESS MACHINES, INC.

918/825-4844

AMERICAN Small Business COMPUTERS

118 SO. MILL ST
PRYOR, OK 74361

Listing continued

```
1390 INPUT"Water Ballast - (Y or N)";W$:IF LEFT$(W$,1)="N" THEN
W=6
1400 A=20:VS=25+2.5*W:C=1/80:IF C$="1" THEN C=1/90:A=23
1410 T=0:H=3280:U=120:D=50:IF C$="O" THEN C=.01:A=26
1420 A1=.34*C/W:B=9510*W/A:GOSUB150
1430 INPUT"Start Speed - Knot";U$:IF U$<>"" THEN U=VAL(U$)
1440 V=U:IF U<VS OR U>130 THEN PRINT"ILLEGAL SPEED":GOTO 1430
1450 GOSUB 150:INPUT"Start Height - Ft";H$:IF H$<>"" THEN H=VAL(
H$)
1460 GOSUB 150:INPUT"Distance-to-go - N.M";D$:IF D$<>"" THEN D=V
AL(D$)
1470 RANDOM:N=RND(0):DS=D:GOSUB 150:IF M$="A" THEN Y=Y+3
1480 NS=N:HS=H:VC=V:WS=W:E=V:PI=3.14159:T=0:U=VC
1490 FOR NA=0 TO 99:PRINT:PRINT"Pilot";NA+1;:INPUT"Name";NA$(NA)

    :IF NA$(NA)="A" THEN M$="A":NA$(NA)="GEORGE" ELSE M$=""
1500 CLS:B$="### ### #### ### ####":A1=.34*C/W:B=9510*W/A:VS=25+
2.5*W
1510 '
    **** provide sky ****

1520 FORZ=0TO4:PRINT@(960-192*Z),USING"# ";2*Z;:PRINTZZ$;:NEXT
1530 PRINT@962,STRING$(61,CHR$(140));:FORI=3TO46:SET(3,I):SET(12
5,I):NEXT
1540 '
    **** and instrument panel ****

1550 PRINT@64,"MACREADY SETTING  =      FPM.";:I=INT(4*K+5*K*N)
1560 PRINT@0,"CONVECTION HEIGHT =";I;" FT.       D   V   H   T
  L/S  ";
1570 PRINT@767,"G";:PRINT@831,"O";:PRINT@895,"A";:PRINT@959,"L";

    :GOTO170
1580 '
     **** another page subroutine ****

1590 PRINT:PRINT"Another page?";
1600 Z$=INKEY$:IF Z$="N" THEN G=-1:RETURN ELSE IF Z$="" GOTO
     1600 ELSE G=0:RETURN
1610 '
     **** COMMANDS routine ****

1620 CLS:PRINTTAB(20)"AVAILABLE COMMANDS":PRINT
1630 PRINT"A - Again, no change in pilot"
1640 PRINT"C - Change atmosphere"
1650 PRINT"D - New day"
1660 PRINT"E - Exit program"
1670 PRINT"H - Help, display available COMMANDs"
1680 PRINT"M - Refer to Flight Manual"
1690 PRINT"N - Next pilot"
1700 PRINT"S - Scoreboard display":PRINT:PRINT"COMMAND?";
1710 CC$=INKEY$:IF CC$="" THEN 1710
1720 IF CC$="S" GOTO 1820
1730 IF CC$="D" THEN NA=99:NEXT:GOTO 1350
1740 IF CC$="M" GOSUB 890
1750 IF CC$="C" THEN NA=99:GOSUB 1880:NEXT:GOTO 1470
1760 IF CC$="H" GOTO 1620
1770 IF CC$="N" GOSUB 1880:NEXT
1780 IF CC$="A" GOSUB 1880:GOTO 1500
1790 IF CC$="E" THEN END
1800 PRINT:PRINT"INVALID COMMAND":FOR Z=0 TO 99:NEXT:GOTO 1620
1810 '
      ***** scoreboard *****

1820 CLS:PRINTTAB(22)"SCOREBOARD":PRINT
1830 PRINT    "PILOT              FIN HT. DISTANCE  TIME  SPEED  P
LACE":PRINT
1840 FORZ=0TONA
1850 PRINTUSING"%                  % #####    #####  ####.# ####.# ##
##";NA$(Z);H(Z);D(Z);T(Z);V(Z);Z+1
1860 NEXTZ:PRINT:PRINT"COMMAND?";
1870 CC$=INKEY$:IF CC$="" THEN 1870 ELSE PRINT:GOTO 1720
1880 T=0:N=NS:L=888:D=DS:V=VC:U=V:W=WS:VS=25+2.5*W:H=HS:RETURN
1890 '
      ***** sort routine *****

1900 SW=0:PS=PS+1
1910 FOR J=0 TO NA:Z=J+1:IF D(J)+V(J)>D(Z)+V(Z) GOTO 1950
1920 ZF$=NA$(J):DZ=D(J):TZ=T(J):VZ=V(J):HZ=H(J)
1930 NA$(J)=NA$(Z):D(J)=D(Z):T(J)=T(Z):V(J)=V(Z):H(J)=H(Z)
1940 NA$(Z)=ZF$:D(Z)=DZ:T(Z)=TZ:V(Z)=VZ:H(Z)=HZ:SW=1
1950 NEXT J:IF SW=1 GOTO 1900
1960 RETURN
1970 '
     ***** data for sound generator *****

1980 DATA 221,33,0,0,17,0,0,1,255,255,213,225,62,1,211,255,9,
     56,252,213,225,62,2,211,255,9,56,252,221,9,56,231,201
```

Catalog Your Files

by Jane Goodale

You need not hassle with filespecs any longer. Instead, install this cataloging routine in programs that maintain sequential filespecs.

I find it very irritating to enter a filespec every time I want to read or write a file. It is much easier to have the filespec in the program. However, it is also very dangerous when running an update program on a sequential file—that is, read a record, change it and write it back under the same filespec.

To avoid this hassle, I chose to install a cataloging routine in all my programs which maintain sequential files. The basic idea is from IBM's technique of generation data sets, though not nearly as elaborate. The procedure requires a driver program which creates and manipulates the catalog, and a standard subroutine in every program using the files to keep the catalog updated and provide the current file names.

The driver program (also used as a menu program) first attempts to read a driver file always called DRIVERXX/CAT; if not found by DOS it will guide you through creating one. Once the driver file is created, the programs keep track of the names of the current files and the next one to create.

The driver file I use is a one-record file containing the system identifier, number of files in the system, and for each file in the system, the file name and generation, date created, and the last program to issue that file. It has the format shown in Table 1.

If the system identifier is INV and there will be two files in the system called DETAIL on drive 1 and SUMARY on drive 2, created by PROGRAMA and PROGRAMB respectively, the filespecs would be:

DETAIL01/INV:1	for the first time created
SUMARY01/INV:2	for the first time created

The driver file would look like:

```
INV2DETAIL01111/20/81
PROGRAMASUMARY01201/15/82
PROGRAMB
```

if the detail file was created Nov. 20, 1981 and the summary file created Jan. 15, 1982.

As time passes and you use the files, the names would become something like this after 15 details and 10 summaries have been created:

```
DETAIL15/INV:1
SUMARY10/INV:2
```

and the driver file:

```
INV2DETAIL15102/15/82
PROGRAMA SUMARY10202/13/82
PROGRAMB
```

Program Listing 1, Catalogs, displays the menu for the applications to follow. It also contains an uncatalog feature which allows you to restart at a previous generation of the file you uncatalog. Program Listings 2 and 3 (called PROGRAMA and PROGRAMB) illustrate use of the subroutines which maintain the catalog. PROGRAMA is fully commented in the standard subroutine. PROGRAMB contains the working parts, but no comments within the subroutine.

To try out the system, key in Catalogs and PROGRAMA. PROGRAMB can be created by modifying PROGRAMA. Save them as CATALOGS/BAS, PROGRAMA/BAS and PROGRAMB /BAS. Be sure to set TIME$ if you want your files dated. Then put formatted disks in the drives you expect to use and run CATALOGS/BAS. Your system will try to find the driver file (DRIVERXX/CAT); when it can't locate it the On Error routine at line 480 will take you to the routine to establish the catalog at line 170.

The screen will describe briefly how

Position	Definition
1–3	Three-character system identifier
4	Number of files for the system (max = 9)
5–10	Six-character file name
11–12	Two character generation number
13	Drive number assigned
14–21	Date last updated
22–29	Name of program which last issued the file
30–54	Repeat of 5–29 for the second file
55–79	Repeat again for the third file
etc.	Up to nine files

Table 1

```
                         ESTABLISH CATALOG
A CONTROL FILE MUST BE BUILT THE FIRST TIME THE PROGRAMS ARE RUN.
ONCE BUILT, IT WILL CONTROL FILE NAMES, DATES, ETC. AND WILL NOT
NEED TO BE ESTABLISHED AGAIN.
NAMING OF FILES WILL BE AS FOLLOWS:
         DRIVERXX/CAT        —DRIVER FILE
         NNNNNNGG/SSS:D      —DETAIL FILE
WHERE:   NNNNNN IS A 6-CHARACTER USER-ASSIGNED NAME
         GG IS THE GENERATION #
         SSS IS THE SYSTEM IDENTIFIER
         D IS A DRIVE NUMBER ASSIGNED TO THE FILES (0-3)
GG WILL START AT 01 AND BE INCREMENTED BY ONE AUTOMATICALLY.
WHEN GG REACHES 99 IT WILL BE RESTARTED AT 01
```

Figure 1

The Key Box

Model I or III
32K RAM
Disk Basic
One Disk Drive
TRSDOS or NEWDOS

```
1 REM ---- CATALOGS/BAS      JANE N GOODALE
2 REM                       828 NO 121 STREET
3 REM                       OMAMA, NE 68154
1Ø CLEAR15ØØ:DEFINTA-Z:CLS:PRINT@47Ø,"SYSTEM DRIVER"

2Ø ONERRORGOTO48Ø
3Ø GOTO15Ø
31 REM ----------------SUBROUTINES------------------------
32 REM ----MISC
4Ø PRINT@99Ø," ";:INPUT"HIT <ENTER> TO CONTINUE";B$;:RETURN
5Ø A$=INKEY$:IFA$=""THEN5ØELSEPRINTA$;:RETURN
6Ø PRINT@99Ø,"ANY KEY TO RETURN ";:GOSUB5Ø:RETURN
61 REM ----CATALOG STATUS
7Ø CLS:PRINTTAB(18)"FILE STATUS FOR SYSTEM ";S$:PRINTTAB(12)STRI
NG$(4Ø,".")
8Ø PRINTTAB(12)"#  FILESPEC";TAB(32)"LAST PGM   DATE":PRINTTAB(1
2)STRING$(4Ø,".")
9Ø FORN=1TOF:DX$=DX$(N)
1ØØ N$=LEFT$(DX$,8):D$=MID$(DX$,9,1):DT$=MID$(DX$,1Ø,8):PG$=RIGH
T$(DX$,8)
11Ø PRINTTAB(11)N;TAB(15)N$+"/"+S$+":"+D$;TAB(32)PG$;TAB(43)DT$
12Ø NEXTN:PRINTTAB(12)STRING$(13,"-");" END OF FILES ";STRING$(I
3,"-"):RETURN
129 REM----SET DX$(N)
13Ø FORN=1TOF:DX$(N)=LEFT$(DF$,25):IFN<FTHENDF$=MID$(DF$,26)
14Ø NEXTN:RETURN
149 REM --------------START-------------------------------
15Ø OPEN"I",1,"DRIVERXX/CAT"
16Ø INPUT#1,DF$:S$=LEFT$(DF$,3):F$=MID$(DF$,4,1):F=VAL(F$):DF$=M
ID$(DF$,5):GOSUB13Ø:GOTO32Ø
169 REM ---------------ESTABLISH CATALOG-----------------
17Ø CLS:PRINTTAB(2Ø)"ESTABLISH CATALOG":PRINT
18Ø PRINT"   A CONTROL FILE MUST BE BUILT THE FIRST TIME THE PRO
GRAMS":PRINT"ARE RUN.  ONCE BUILT, IT WILL CONTROL FILE NAMES, D
ATES, ETC.":PRINT"AND WILL NOT NEED TO BE EASTABLISED AGAIN."
19Ø PRINT"NAMING OF FILES WILL BE AS FOLLOWS :":PRINTTAB(8)"DRIV
ERXX/CAT    -  DRIVER FILE":PRINTTAB(8)"NNNNNNGG/SSS:D  -  DETAI
L FILE":PRINT"WHERE : NNNNNN IS A 6 CHARACTER USER ASSIGNED NAME
"
2ØØ PRINTTAB(8)"GG IS THE GENERATION #":PRINTTAB(8)"SSS IS THE S
YSTEM IDENTIFIER":PRINTTAB(8)"D IS A DRIVE NUMBER ASSIGNED TO TH
E FILES (Ø-3)":PRINT"GG SILL START AT Ø1 AND BE INCREMENTED BY 1
 AUTOMATICALLY."
21Ø PRINT"WHEN GG REACHES 99 IT WILL BE RESTARTED AT Ø1":GOSUB4Ø
22Ø CLS:PRINTTAB(2Ø)"FILE DEFINITION":PRINT
23Ø INPUT"ENTER 3 CHAR SYSTEM ID";S$
24Ø IFLEN(S$)=3THEN25ØELSEPRINT"** 3 CHAR ID PLEASE":GOTO23Ø
25Ø INPUT"ENTER NUMBER OF FILES FOR THE SYSTEM ( 1 TO 9 )";F$
26Ø IF(F$<"1")OR(F$>"9")THENPRINT"** 1 TO 9 PLEASE":GOTO25Ø
27Ø DF$=S$+F$:F=VAL(F$):PRINT
28Ø FORN=1TOF:PRINT"FILE # ";N;"DEFINITION"
29Ø INPUT"ENTER 6 CHAR NAME";N$:IFLEN(N$)=6THEN3ØØELSEPRINT"** 6
 CHAR NAME PLEASE":GOTO29Ø
3ØØ INPUT"ENTER DRIVE #";D$:IF(D$>="Ø")AND(D$<"4")THEN31ØELSEPRI
NT"** DRIVE # Ø TO 3 PLEASE":GOTO3ØØ
31Ø DF$=DF$+N$+"ØØ"+D$+LEFT$(TIME$,8)+"CATALOGS":FS$=N$+"ØØ"+"/"
+S$+":"+D$:OPEN"O",2,FS$:PRINT#2,"DUMMY"+FS$:CLOSE2:PRINT:NEXTN:
OPEN"O",1,"DRIVERXX/CAT":PRINT#1,DF$:DF$=MID$(DF$,5):GOSUB13Ø:GO
TO32Ø
319 REM ------------------MAIN MENU---------------------------

32Ø CLS:PRINT"SELECT ONE OF THE FOLLOWING :":PRINT
33Ø PRINTTAB(8)"1  PRINT CATALOG STATUS":PRINTTAB(8)"2  UNCATALO
G A FILE":PRINTTAB(8)"3  RUN PROGRAM A":PRINTTAB(8)"4  RUN PROGR
AM B":PRINTTAB(8)"5  RETURN TO SYSTEM COMMAND"
34Ø PRINT:PRINT"CHOICE (DO NOT HIT <ENTER>)";:GOSUB5Ø
35Ø ONVAL(A$)GOTO39Ø,41Ø,37Ø,38Ø,4ØØ
36Ø PRINT"ILLEGAL RESPONSE":GOSUB6Ø:GOTO32Ø
369 REM ------------------CHOICES---------------------------
37Ø CLOSE1:RUN"PROGRAMA/BAS"
38Ø CLOSE1:RUN"PROGRAMB/BAS"
39Ø GOSUB7Ø:GOSUB6Ø:GOTO32Ø
4ØØ CLOSE1:CMD"S"
409 REM ----------------UNCATALOG-------------------------
41Ø GOSUB7Ø:PRINT@181,"UNCATALOG";:PRINT@247,"Y/N";
42Ø FORN=1TOF:L=55+(N+3)*64
425 PRINT@L,"-";:GOSUB5Ø:IF(A$="Y")OR(A$="N")THEN43ØELSEPRINT@L+
1," ";:GOTO425
43Ø IFA$="Y"THENGOSUB45Ø
44Ø NEXTN:CLOSE1:OPEN"O",1,"DRIVERXX/CAT":DF$=S$+F$:FORN=1TOF:DF
$=DF$+DX$(N):NEXTN:PRINT#1,DF$:CLOSE1:GOTO32Ø
45Ø G$=MID$(DX$(N),7,2):G=VAL(G$):IFG>ØTHENG=G-1
46Ø G$=STR$(G):G$=MID$(G$,2):IFG<1ØTHENG$="Ø"+G$
47Ø DX$(N)=LEFT$(DX$(N),6)+G$+MID$(DX$(N),9,1)+LEFT$(TIME$,8)+"C
ATALOGS":RETURN
479 REM -----------------ON ERROR ROUTINE-----------------
48Ø IFERL=15ØCLOSE1:RESUME17Ø
49Ø PRINT"ERROR ";ERR/2+1;" IN LINE ";ERL:END
```

Program Listing 1. Catalogs

```
1 REM ----     PROGRAMA/BAS        JANE N GOODALE
2 REM                             828 NO 121 STREET
3 REM                             OMAHA, NE 68154
1Ø CLEAR3ØØØ:DEFINTA-Z:CLS:PRINT@47Ø,"RUNNING PROGRAM A":GOTO2ØØ

8Ø REM-----------I/O SUBROUTINE TO MAINTAIN CATALOG-----------
81 REM
82 REM          ------------VARIABLES---------------
83 REM          ZR$       - DRIVER RECORD
84 REM          ZS$       - 3 CHAR SYSTEM IDENTIFIER
85 REM          ZF$       - STRING VALUE OF # OF FILES
86 REM          ZF%       - NUMERIC VALUE OF # OF FILES
87 REM          ZX$       - TEMPORARY HOLDING AREA FOR ZR$
88 REM          ZZ%       - FOR/NEXT VARIABLE: POINTER TO FILE NAM
ES
89 REM          ZR$(ZZ%) - ARRAY OF EACH 25 CHAR DEFINITION
9Ø REM          ZG$       - STRING VALUE OF GENERATION #
91 REM          ZG%       - NUMERIC VALUE OF GENERATION #
92 REM          ZI$(ZZ%) - ARRAY OF INPUT FILESPECS
93 REM          ZO$(ZZ%) - ARRAY OF OUTPUT FILESPECS
94 REM
99 REM---- OPEN DRIVER FILE, READ AND GET INPUT FILE NAMES.
1ØØ OPEN"I",1,"DRIVERXX/CAT":INPUT#1,ZR$:CLOSE1:ZS$=LEFT$(ZR$,3)
:ZF$=MID$(ZR$,4,1):ZF%=VAL(ZF$):ZX$=MID$(ZR$,5)
11Ø FORZZ%=1TOZF%:ZR$(ZZ%)=LEFT$(ZX$,25):ZI$(ZZ%)=LEFT$(ZR$(ZZ%)
,8)+"/"+ZS$+":"+MID$(ZR$(ZZ%),9,I):IFZZ%<ZF%THENZX$=MID$(ZX$,26)
12Ø NEXTZZ%:RETURN
121 REM---- GET NEW GENERATION # FOR OUTPUT AND OPEN OUTPUT
122 REM     FILES.  SET ZZ% TO FILE NUMBER BEFORE GOING INTO
123 REM     SUBROUTINE.
13Ø ZG$=MID$(ZR$(ZZ%),7,2):ZG%=VAL(ZG$)+1:IFZG%>99THENZG%=1
14Ø ZG$=STR$(ZG%):ZG$=MID$(ZG$,2):IFZG%<1ØTHENZG$="Ø"+ZG$
141 REM---- UPDATE FILE # ZZ% PORTION OF DRIVER RECORD.
142 REM     PROGRAM NAME SHOULD BE 8 CHAR IN LENGTH OR PADDED
143 REM     WITH SPACES.
15Ø ZR$(ZZ%)=LEFT$(ZR$(ZZ%),6)+ZG$+MID$(ZR$(ZZ%),9,1)+LEFT$(TIME
$,8)+"PROGRAMA"
16Ø ZO$(ZZ%)=LEFT$(ZI$(ZZ%),6)+ZG$+MID$(ZI$(ZZ%),9):RETURN
161 REM---- AT END OF JOB, WRITE DRIVER FILE BACK OUT IF ANY
162 REM     OUTPUT FILES WERE WRITTEN.
17Ø ZR$=ZS$+ZF$:FORZZ%=1TOZF%:ZR$=ZR$+ZR$(ZZ%):NEXTZZ%:OPEN"O",1
,"DRIVERXX/CAT":PRINT#I,ZR$:CLOSEI:RETURN
195 REM---- DEMONSTRATION PROGRAM WILL INPUT FILE #3
196 REM                              OUTPUT FILE #I AND #3
2ØØ GOSUB1ØØ:OPEN"I",1,ZI$(3):ZZ%=1:GOSUB13Ø:ZZ%=3:GOSUB13Ø:OPEN
"O",2,ZO$(1):OPEN"O",3,ZO$(3)
21Ø INPUT#1,R$:PRINT#3,R$:PRINT#2,R$:PRINT#2,R$:IFEOF(1)THEN22ØE
LSEGOTO21Ø
22Ø CLOSE1,2,3:GOSUB17Ø:CLS:PRINT@47Ø,"END OF PROGRAM A":RUN"CAT
ALOGS/BAS"
```

Program Listing 2. PROGRAMA

the system works (see Fig. 1). The next screen (Fig. 2) will ask for a three character system ID and how many files (different filespecs) you want in the system. Create at least three to use with demonstration program PROGRAMA. You are then prompted to enter a six-character file name and drive number for however many files you have specified. After each response, a dummy file will be generated.

When all files have been described, the catalog is established and you are taken to the main menu, the normal starting point (see Fig. 3). Choose Option 1 to print the catalog status and you will see your files displayed as in Fig. 4.

Return to the menu by hitting any key and make Selection 3 to run PROGRAMA. When it is finished you will be returned to the Catalogs program menu. Select PROGRAMA several more times. For the examples here I ran PROGRAMA three times. Now select Item 1 again to see the catalog status—it should look something like Fig. 5. PROGRAMA writes only files 1 and 3 so the second file shows no change.

Select Option 2 next to uncatalog. Your screen will look like Fig. 6. If you answer Y to file 1 and N to the rest, your next check of the catalog status should look like Fig. 7. The generation number of the first file has been decreased by one.

If you return to DOS at this point and check your directory, you will see that all files created are still there (Fig. 8). The uncatalog does not kill the file. However, the driver file will force your program to read the correct file and recreate the next one in line.

Note that PROGRAMB reads the files, but never writes any. In this case, lines 130–170, the output maintenance subroutines, could be removed entirely.

If your DOS allows you to kill files within a Basic program, logic could be installed to kill a file that is uncata-

```
1 REM ---- PROGRAMB/BAS                JANE N GOODALE
2 REM                                  828 NO 121 STREET
3 REM                                  OMAHA, NE 68154
10 CLEAR3000:DEFINTA-Z:CLS:PRINT@470,"RUNNING PROGRAM B":GOTO200

80 REM -----------I/O SUBROUTINE TO MAINTAIN CATALOG-----------
100 OPEN"I",1,"DRIVERXX/CAT":INPUT#1,ZR$:CLOSE1:ZS$=LEFT$(ZR$,3)
:ZF$=MID$(ZR$,4,1):ZF%=VAL(ZF$):ZX$=MID$(ZR$,5)
110 FORZZ%=1TOZF%:ZR$(ZZ%)=LEFT$(ZX$,25):ZI$(ZZ%)=LEFT$(ZR$(ZZ%)
,8)+"/"+ZS$+":"+MID$(ZR$(ZZ%),9,1):IFZZ%<ZF%THENZX$=MID$(ZX$,26)

120 NEXTZZ%:RETURN
130 ZG$=MID$(ZR$(ZZ%),7,2):ZG%=VAL(ZG$)+1:IFZG%>99THENZG%=1
140 ZG$=STR$(ZG%):ZG$=MID$(ZG$,2):IFZG%<10THENZG$="0"+ZG$
150 ZR$(ZZ%)=LEFT$(ZR$(ZZ%),6)+ZG$+MID$(ZR$(ZZ%),9,1)+LEFT$(TIME
$,8)+"PROGRAMB"
160 ZO$(ZZ%)=LEFT$(ZI$(ZZ%),6)+ZG$+MID$(ZI$(ZZ%),9):RETURN
170 ZR$=ZS$+ZF$:FORZZ%=1TOZF%:ZR$=ZR$+ZR$(ZZ%):NEXTZZ%:OPEN"O",1
,"DRIVERXX/CAT":PRINT#1,ZR$:CLOSE1:RETURN
195 REM ------DEMONSTRATION PROGRAM WILL READ FILES 1 AND 3
196 REM       AND DISPLAY THEM.
200 GOSUB100:OPEN"I",1,ZI$(1):OPEN"I",2,ZI$(3):CLS:PRINT"FILE 1
FOLLOWS":PRINT
210 INPUT#1,R$:PRINTR$:IFEOF(1)THEN220ELSEGOTO210
220 PRINT:PRINT"FILE 3 FOLLOWS":PRINT
230 INPUT#2,R$:PRINTR$:IFEOF(2)THEN240ELSEGOTO230
240 CLOSE1,2:PRINT:PRINT"END OF PROGRAM B"
250 INPUT"<ENTER> TO CONTINUE";A$:RUN"CATALOGS/BAS"
```

Program Listing 3. PROGRAMB

```
FILE DEFINITION
ENTER 3 CHAR SYSTEM ID? INV
ENTER NUMBER OF FILES FOR THE SYSTEM (1 TO 9)? 3

FILE # 1 DEFINITION
ENTER 6 CHAR NAME? DETAIL
ENTER DRIVE #? 1

FILE # 2 DEFINITION
ENTER 6 CHAR NAME? INPUT1
ENTER DRIVE #? 2

FILE # 3 DEFINITION
ENTER 6 CHAR NAME? SUMARY
ENTER DRIVE #? 1
```

Figure 2

```
                 FILE STATUS FOR SYSTEM INV

#   FILESPEC          LAST PGM    DATE       UNCATALOG
                                             Y/N
1   DETAIL03/INV:1    PROGRAMA    11/15/81   -Y
2   INPUT100/INV:2    CATALOGS    11/15/81   -N
3   SUMARY03/INV:1    PROGRAMA    11/15/81   -
                     END OF FILES
```

Figure 6

```
           FILE STATUS FOR SYSTEM INV

#   FILESPEC          LAST PGM    DATE

1   DETAIL02/INV:1    CATALOGS    11/15/81
2   INPUT100/INV:2    CATALOGS    11/15/81
3   SUMARY03/INV:1    PROGRAMA    11/15/81
               END OF FILES


           ANY KEY TO RETURN
```

Figure 7

```
DRIVE   1   DATA0003   11/15/81   39 TRKS   53 FDES   128 GRANS

DRIVERXX/CAT   SUMARY00/INV   SUMARY01/INV   SUMARY02/INV
SUMARY03/INV   DETAIL00/INV   DETAIL01/INV   DETAIL02/INV
DETAIL03/INV

DRIVE   2   DATA0003   11/15/81   39 TRKS   61 FDES   129 GRANS

INPUT100/INV
```

Figure 8

```
SELECT ONE OF THE FOLLOWING:

  1 PRINT CATALOG STATUS
  2 UNCATALOG A FILE
  3 RUN PROGRAM A
  4 RUN PROGRAM B
  5 RETURN TO SYSTEM COMMAND

CHOICE (DO NOT HIT <ENTER>)
```

Figure 3

```
           FILE STATUS FOR SYSTEM INV

#   FILESPEC          LAST PGM    DATE

1   DETAIL00/INV:1    CATALOGS    11/15/81
2   INPUT100/INV:2    CATALOGS    11/15/81
3   SUMARY00/INV:1    CATALOGS    11/15/81
               END OF FILES

           ANY KEY TO RETURN
```

Figure 4

```
           FILE STATUS FOR SYSTEM INV

#   FILESPEC          LAST PGM    DATE

1   DETAIL03/INV:1    PROGRAMA    11/15/81
2   INPUT100/INV:2    CATALOGS    11/15/81
3   SUMARY03/INV:1    PROGRAMA    11/15/81
               END OF FILES


           ANY KEY TO RETURN
```

Figure 5

loged and kill every file over N generations old.

Trying to uncatalog the 00 version of all files will not restart the system from scratch. In fact, it will do nothing at all. To restart the system, simply kill DRIVER XX/CAT.

Using this system requires that you have one and only one system's data on your data disk. It cannot now handle more than one catalog on the same disk since the catalog, which is the driver file, is called DRIVERXX/ CAT in every case.

A more elaborate cataloging system could be developed to kill files and maintain unique driver files for each system, with names such as DRIVEINV/CAT, DRIVEPAY/CAT and DRIVEREC/CAT. The number of files per system could be unlimited by using a driver file of more than one record.

However, the maintenance program and subroutines would be much more complicated. As they are presented here, if compressed and with all comments removed, Catalogs takes 2605 bytes of memory and the I/O subroutine takes 503 bytes of memory.

I keep a raw copy of each of these available for use in any application program systems that will do much file manipulation. The application program is started at line 200 of PROGRAMA or PROGRAMB. Install your own menu driver in Catalogs by changing line 330 and lines 370–380.

So what does all this mean? A file you want to call DETAIL../INV can be called DETAIL00/ INV when the catalog is initialized. More importantly, it will be DETAIL01/INV for the first run, DETAIL02/INV for the second run, DETAIL03/INV for the third run and so on. No operator intervention is necessary once the catalog is established. It also means you always have a file to go back to if you goof the current run, by using the uncatalog feature. ■

Jane Goodale (828 N. 121 St., Omaha, NE 68154) is employed by Western Electric.

Directory Information, Please

by Charles Knight

No disk directory can suit everyone's needs. But this one, written for LDOS, has enough flexibility to be adapted with relative ease.

Almost on Day 1, the average disk-drive owner finds he needs to catalog his programs and data files. Many programs do the job, but none of them can be all-inclusive.

They range from simple programs that store only a program name and disk number to more elaborate programs that let you add to the file. Some use a simple CMD"DIR" statement and PEEK the filespecs from the screen, severely limiting their usefulness, while others use machine-language disk-read routines and require only a single-density disk with its directory on track 17.

All such disk catalog programs that I've seen use memory to store the information and write the file to disk after sorting and processing the file in memory. This method is fast, but has one major disadvantage: it only lets you store a fixed number of file names at a time. If you have too many, you must break your catalog up into several smaller files. If you use a printout to locate a file, you can spend quite some time to find which of a group of printouts contains the entry for the desired file. You also can't create one large sorted printout of all of your files.

I wrote this database management system to get around this limitation. I did have to make a trade-off though—because the program can deal with a database much larger than available memory, the data files will take a long time to sort. This is a problem inherent in any hardware design using 5¼-inch floppies, since data must be moved around physically on the disk and disk-access speed is limited. A 5,000-record file can take overnight to sort.

But most of us do sleep and go to work, and you can use these times. Also, most databases don't need constant sorting. I sort at three- to six-month intervals.

And if your database is small (under 800 items), the sort time is only a few seconds.

Requirements

The program had to read and catalog all of my disks whether they were single- or double-density regardless of what operating system created them. I couldn't accommodate the Percom DBLDOS format (which is also the NEWDOS80 standard), but I don't use these two much, anyway. I use LDOS almost exclusively.

I wanted to know at a glance not only which disk my file was on, but also something about that file. Since I often have several versions of a similarly named program, I needed to know the file size and attributes to identify the file. Also, I wanted to know whether the disk it was on was single- or double-density, and whether that disk had any free space left. In addition, I wanted to print labels for the disks as they were read in by the system so I could label them promptly—before I got them out of order again.

LDOS offered locations in memory where all the information I wanted was kept, and it was a simple matter to extract it and place it within my database. Since I do use an occasional single-density disk formatted under DOSPLUS, NEWDOS80 version 1, and NEWDOS 2.1, I wanted to have my database program handle its repair to LDOS readable status for me.

The one common feature I didn't need was the ability to add user comments about the file, since I'm too lazy to use such a feature anyway.

The database system I came up with consists of these program modules:

- MAGIDEX/CMD—Machine language convenience loader.
- MAGIDEX/BAS—Menu program to call up the other modules.
- DREADER/BAS—Directory reader and database builder.
- SORT/BAS—Sort program for LDOS 5.1.0 and later.
- SORT50/BAS—Sort program for LDOS 5.0.3 and earlier.
- PRINT/BAS—Module for printing out the database.
- UTILITY/BAS—Module for changing the database defaults.

MAGIDEX/CMD is a convenience loader for the rest of the system. You can enter LBasic and simply run MAGIDEX, but that requires more typing and lacks elegance. The source code here is in EDAS format. If you use EDTASM, you'll have to put each graphics character on a separate line with its own

DEFB statement. This loader uses a jump vector at X'4405' that will load and execute a load file format program. LBasic, of course, is such a program. This vector is in TRSDOS also, but its use is severely limited for Basic programs by those annoying and rarely needed HOW MANY FILES and MEMORY SIZE messages. By pointing the HL register pair to a buffer that contains the command to be executed, followed by a carriage return and a jump to this vector, the command in the buffer will be executed along with any parameters that happen to be with it.

This loader module also displays some pleasing graphics while LBasic is loading and executing. You don't have to type in this machine-language loader, but you'll save time and have a more professional-looking software package.

MAGIDEX/BAS serves as a menu from which the other modules are called, and ties them together into a complete database-management system. It first checks to see if the system has been initialized by searching for a data file named TRANSFER/DAT, which contains all the data items that must be passed from program to program. If TRANSFER/DAT isn't there, MAGIDEX/BAS will assume that the system has not been initialized.

Much of the screen is devoted to a dynamic graphics presentation. This assures you that the machine is still running, and improves the program's appearance. The menu will also display your name, the number of disks, and the number of files currently in the system. If you haven't built your files yet, it'll tell you that instead.

You're given four options. The first is "Enter more disks," which you select if you're running the program for the first time or if you've got some new programs to add. It calls the DREADER/BAS module and sets up the system to read the file names and other disk data from your disks.

The second is "Sort data." This module checks which version of LDOS you're running and calls the proper sort module.

The third option is "Print Disk Data," which loads the printout module that searches the database and prints out the information you've catalogued.

The final option, "End Program," returns you to LDOS command level after displaying a short message. You should always exit MAGIDEX this way to be sure that all files are closed.

DREADER/BAS

The DREADER/BAS module is the heart of the system. It reads in the disks one at a time, assigns each one a disk number, and builds the database. It also lets you select certain options. If this is the first time you've run the program, or if you're starting a new database, you'll be asked your name, which will become a part of the system and will be used on the printouts and in the menu program.

Next, you're asked if you want to include invisible files. Normally, the only ones you'll have are those that are part of your operating system; you won't want to include them, so answer "N." (You can answer yes or no questions in either upper- or lowercase.)

Now you enter the number of the disk drive you want to use to hold your files. You *must* leave the disk in that drive while you're running the database, or you might lose all your data. You can change drives only by using the UTILITY/BAS program described later.

Next, you're asked if you want to print disk labels as the disks are read in. If you answer "Y," be sure your printer is loaded with standard mailing labels such as those sold by Radio Shack. The information printed on the label will be the disk name and date, amount of free space in both grans and K, disk size and number of sides, single- or double-density, and the number of cylinders on the disk.

Now (and each time the system has finished with a disk) you'll be prompted "Enter the drive containing the disk to be read:" and a blinking question mark for a cursor. If you're finished cataloging disks, answer with "@," and the system will close files and return you to the menu. Do this periodically to be sure that a parity error or some other malfunction doesn't leave you with an unusable set of data files, and a greatly expanded vocabulary as well.

Suppose you answer with "2." At this point, the program will begin reading the GAT sector of the disk in drive 2 to determine its name, date, and amount of free space. This information will be printed at the top of the screen.

Next, the program will open the directory of the target disk as a random file and begin to read in all files, discarding those that have been killed or are system files. As each file is catalogued, it is displayed on the screen, together with its attributes and its size. Two files show on the screen at one time, so if you have a speed-up modification, you'll have to be a fast reader to keep up. When the program finishes reading the disk, you'll get the prompt again to enter a drive number.

If for any reason LDOS or LBasic can't open the file DIR/SYS on the target disk, you'll receive the prompt "Directory not readable by LDOS, do you wish to attempt to repair it?" If you answer "N," you'll again be prompted to enter the drive number. If you answer "Y," the file REPAIR/CMD, which comes with all versions of LDOS past 5.0.2, must be on one of the disks in the system—preferably in drive 0. This program is then executed; if the repair is successful, the disk can now be read. If not, you'll be told so and returned to the drive number prompt. No harm is done to a disk that cannot be repaired.

If the disk has anything wrong that makes its directory unreadable, you'll reach this part of the program. This option is necessary because TRSDOS and NEWDOS 2.1 write an incorrect hash code for the file DIR/SYS, and the operating system can't find that file. It's worth noting that the hash code for BASIC/CMD on a TRSDOS disk is also incorrect, but this one does work. Under LDOS, the directory data address mark has nothing whatever to do with its ability to read the directory from LBasic as though it were a random data file.

While you're running the DREADER/BAS module, you may want to have MONITOR/CMD from Misosys resident to intercept disk errors and give you a chance to repeat a sector write in the event of an error. This will mess up the nice graphics on the screen, but it may save your data files by giving you second and third tries at a sector write. I also recommend running with SYS1 through SYS3 and SYS8 and SYS10 resident (LDOS 5.1 only), to speed up the disk reading operation. You'll have to get them back out of memory, though, before you try to sort the file.

You will suffer a total program crash if you get a disk I/O error while writing the file TRANSFER/DAT after you've read in all your disks and requested a return to the menu; the program will hang in an error-trap loop. Since TRANSFER/DAT is written to just before returning to the menu, the odds of such a crash happening are slim, so I haven't written code for this possibility.

Manipulating the Data

SORT/BAS and SORT50/BAS are the data-manipulating hands of the system. After you've read in all your disks, it's time to sort your data file into alphabetical order. As mentioned above, if you have more than 800 filespecs in your file, you should allow quite a

while. A 1,000-record file will take about two hours to sort; a 5,000-record file will probably require overnight. The number of disks you have doesn't matter. Only the number of filespecs is important.

I've got some 2,800 records in my library, and it takes about six hours to sort them. I store the records on a single-density system disk, so you can imagine the total capacity of the system.

The sort can be postponed for now and the printout module will still work, but the printout won't be sorted. In fact, it will be a printout by disk in the order in which they were read. You can still use the sequential search features in the system, though. If the file is less than 800 records, the entire file is brought into memory for the sort. In either event, you should back up the files before trying to sort them because of the large amount of disk I/O going on. The module for LDOS 5.0 uses a shell-sort algorithm and the LSET statement to by-pass Microsoft's garbage-collection routines. Because all of the strings being sorted are the same length (32 bytes), this LSET technique is the fastest, and it's a whole lot easier than using VARPTR and POKE. If you're using LDOS 5.1, the records are pulled into memory and the CMD"O" machine-language sort of LBasic is used.

The versatility built into these sort modules keeps you in control of the tradeoffs necessary due to memory limitation. If you restrict yourself to short files as other catalog programs require, the sort will be quick; if you prefer a larger database, you can have that as well. You can't quite have your cake and eat it too, but you've got more flexibility than with most software.

PRINT/BAS

Think of PRINT/BAS as the mouth of the database system. From this module, you get hard copy and video. The ultimate function of any database-management system is the printout. This module provides several options.

Option 1—Print file to printer. This sends the entire file to the printer in the order in which it is stored on the disk. (Either as the files were read or alphabetically, depending on whether you've sorted the file.) You're asked for a starting and an ending record number. If you press enter twice, you send the entire file to the printer. The output is formatted to print nine filespecs per page, and each filespec is outdented from the rest of the information for easy reference. You can print the entire file once and keep the information in a binder.

This printout contains nearly all the information in the directory about the file except its location on the disk. Of course, the inventive programmer can add that, too, but I don't need that much information. Each page is numbered and has a title header printed at its top. You can select the short-form printout option to save paper or if you have a slow printer, but the printout won't contain nearly as much information about the files. The long-form printout is the reason for this program and is what makes it unique among directory management systems.

Option 2—Search for a specific file. This option lets you search for a particular filespec, partspec, or extension. It is an instring sequential search.

You're asked to enter the filespec or partspec to be searched for. For example, if you input /BAS, you'll find all files that have /BAS as an extension. BAS finds BASCOM, HELP/BAS, and BASIC/CMD. Use only uppercase, since that's how the filespecs are stored.

Then you're asked if you want output to your printer. Any input other than "Y" or "y" outputs to the screen only. Each file is displayed on the screen; if you don't specify printer output, you're prompted to press enter for the next record. Should the search not find anything, you're automatically returned to the menu. Whenever the flashing "enter" appears on the screen, you can press the up arrow instead of enter and return to the print module menu. Most printouts also abort to the menu if you hold the up arrow down. Disk I/O and other errors return to the menu too, but you're told that first and asked to press the enter key.

Option 3—Printout of all files to the screen. This is the same as option 1 except that output is to the video only. Pressing enter displays the next record; pressing the up arrow aborts back to the print module menu. The lower part of the screen clears just before the display of each file.

Option 4—Return to master menu. This is the proper way to exit the print module and return to the database master-menu. Note that the number 4 is used in both the print module and the menu for the exit option (making it easier to remember).

Option 5—Printout of header data. This option prints out information about the disks only, not about the files on them. The printout is such that you can cut it out and file it with or on the disk jacket.

Option 6—Printout of files by disk. This is a sequential search and printout routine. It asks for the starting and ending disk number; pressing enter twice here defaults to print the entire file. Each record comes from the HEADER/DAT file, and the DIRECTRY/DAT file is searched for files that match this disk number. The output is directed to the screen or both printer and screen. The defaults output the entire file to the screen only.

Since this sequentially searches the whole file once for each disk, it can take quite a while to perform for all disks on a very large file. If you're making a printout, you won't mind taking time out for a Coke or watching something on tv while the computer does the dirty work for you.

All of the program modules include a routine used on initialization. You must determine if a file called TRANSFER/DAT exists. If not, some of the modules return to the menu. In the case of the menu itself, it displays certain messages to note that the system is not yet initialized. In the MAGIDEX/BAS program, this routine is at lines 160 to 280. By placing the open statement after an ON ERROR GOTO statement, you can determine the existence of the file TRANSFER/DAT and take appropriate action.

The routine for blinking all the graphics blocks in MAGIDEX/BAS is a loop at lines 720 to 820. This loop executes until you press a key between 1 and 4. Each time the loop executes, the value of CK increases by two. If that value exceeds 30, it sets to 5 to keep the flashing graphics bars within the limits of the screen. The PEEK in line 880 checks the OSVER$ within LDOS to determine which of the two sort modules should be loaded.

In SORT/BAS, the first order of business is to determine the length of the data file. This value is contained within TRANSFER/DAT and is read into the variable IN. Regardless of the actual number of records in DIRECTRY/DAT, the system assumes that the value read in is the length of the file. Thus, you should make sure that you always use the correct TRANSFER/DAT to keep from ruining your database. If you have more than 800 files, the external sort in lines 520 to 720 runs. If not, the routine at 780 to 1020 executes. If you meet an out-of-memory error, the program tells you that the SYSGEN is probably too large and reminds you to reduce it by aborting to line 1040. If you are running under LDOS 5.0, the module SORT50/BAS executes instead of SORT/BAS. The major difference is in lines 820 to 1300. SORT0/BAS has a

shell-sort algorithm. Although you can't run SORT50/BAS under LDOS 5.1, the sort is slightly slower; instead of taking about a minute for 800 files, it takes about six. Since LBasic has the CMD"O" function, it's foolish not to use it. They both perform the same on files longer than 800 records. You may elect not to key in the sort routine you don't need, but MAGIDEX/BAS must have its line 880 modified accordingly.

The DREADER/BAS module is where most of the work is done. The AB$ array is set to the attributes a file can have. Should you ever receive the display of "*****" for a file's attributes, copy it to another disk: its directory entry has been clobbered. Bits 0,1, and 2 of the first byte of a file primary directory entry, or FPDE, contain the attributes of a file. The value thus obtained is used as the subscript to print out the correct text to match the attributes of a file. The same technique is used with MO$ to select the text for the month.

Lines 280 and 300 define two functions that will be called often during program execution. Line 280 removes the right blanks from a filespec that is less than eight characters long, while line 300 defines the function for adding the colon before the drive specification.

The main directory-read loop begins at line 640. First, the file DIR/SYS on the requested drive opens with a logical record length of 1 and the subroutine at line 1680 reads its granule allocation, or GAT table. Then the directory reopens with a logical record length of 32 bytes. As soon as the disk is read, the drive code table, or DCT, contains all the information about density, size, etc. and all you need to do is PEEK it out. Since the DCT is exactly 10 bytes long, just multiply the drive number by 10 and add the result to the start of the DCT to get the DCT for the correct drive.

The DCT for drive 0 begins at 18176, but the first byte you're concerned with is at 18179. This byte contains the density and size of the disk just accessed. Based on the value of the proper bit, SD$ is set to indicate the correct condition in line 680 and D8$ is set in line 700. Bit 5 in the next byte tells you that the disk is two-sided if it is set. The Device command in LDOS reads the DCTs in a similar manner and reports on their status, which LDOS remembers according to the last disk accessed by that drive. By adding 1 to the number PEEKed from 18182, you can see how many cylinders were on the last disk read.

After fielding the buffer in line 820 to match the bytes in the directory you want to read, the program enters a loop beginning at line 860. Line 900 tests bits 4 and 7 to determine that the file hasn't been killed or that the entry is not a file extended directory entry, or FXDE. The next line tests bit 3 and the flag to indicate whether invisible files are to be included and exits if both tests pass. You won't normally want to include system files in the directory, so they're eliminated in line 940. After the program decides that the file is to be included, the filespec is constructed in line 960 by adding a slash after the filespec and before the extension. At this time, two subroutines are called to determine the file's attributes and unpack the data into a format more recognizable by us humans. The logical record length is adjusted in line 1040. This value will be the actual LRL if it is anything but zero, but if it's zero, then the true LRL is 256. By checking the password hash codes in lines 1060 and 1100, the program sees whether the file is password protected and, if so, whether its password is PASSWORD. Of course, if you had all day to read each directory you could decode each password to one that would work, but that isn't the purpose of a disk directory management program.

The code in line 1140 removes the slash mark added earlier if the file has no extension. Then a subroutine is called which calculates the size of the

file both in granules and kilobytes. At this point, after the information appears on the screen, the data is written into the file DIRECTRY/DAT and the procedure repeats for the next directory entry. When the directory has been completely read, the files are closed to update the EOF pointer so that the files won't be completely lost if an I/O error occurs. Then control is passed back to line 540, where the process can be repeated on the next disk in your library.

The subroutine at line 1860 calculates the free space on the disk by reading the GAT table and checking each bit of each byte to determine if the granule it represents is allocated or free. Only the high four bits are checked, since this is all that LDOS uses. In line 2120, the number of free granules is converted into K and the subroutine returns with EE equal to the number of free grans and EK equal to the amount of free space.

The subroutine at line 2180 writes the data to the file DIRECTRY/DAT, which contains information about the files, while the subroutine at line 2340 handles writes to the file HEADER/DAT, which contains information about the disk. As you can see, the relative sizes of these two data files will vary depending on how many files are on each disk. A pointer stored within DIRECTRY/DAT points to the correct data in HEADER/DAT. This is why it's not possible to calculate exactly how many disks may be stored in a given amount of disk space. I'll wager, though, that a double-density 40-track data disk will hold more files than you'll ever have.

The screen graphics routine is next at line 2620. This is called at program initialization; the screen is redrawn each time you call the program REPAIR/CMD, since it clears the screen before executing. The data-input routine is at line 2700. This versatile routine accepts only a predetermined amount of characters, and returns with the input stored in TT$ only after you hit the enter key. Before you call it, S and E must be loaded with the starting and ending points on the screen where input is to be accepted. The length of the input string is the difference between them. The routine takes care of calculating this and keeps the result stored in C.

The file TRANSFER/DAT is written to in the routine at line 3020. The data stored in this file is VI, the flag indicating whether or not the operator wants to include invisible files; D2 and D2$, the drive number that the data files will be written on; DN, the current highest disk number assigned; IN, the number of filespecs currently stored in the system; and KT$, your name. This file will never occupy more than a granule of disk space, but it must be present to let the programs communicate with each other.

The rest of the program contains the error-trapping routines. If an error occurs that the program hasn't planned for, control passes to line 3340, which jumps to 3020, the exit routine. The screen shows the error-code number while the menu program is loading. This procedure assures you that all files are closed and have their directory properly maintained, should recovery be necessary.

I dislike reconstructing data files, so I exit the DREADER/BAS program after every 15 or 20 disks have been read and make a back-up of all three data files.

PRINT/BAS

PRINT/BAS is the longest of the modules. It reads the data files created by DREADER/BAS and outputs them to the screen or printer. After you initialize, control passes to line 980, which displays a menu on the screen. Line 1220 is a loop that prints a moving ar-

row as a prompt. The ON...GOTO statement in line 1360 goes to the proper routine; otherwise, the position of the arrow advances and the loop repeats.

The routines at line 1400 and line 300 are similar. They print the entire file, though part of the file can be printed if you know the record number range. I wrote these routines separately to let you modify the printout format without affecting the video. The code is longer, but more versatile.

Line 2040 provides pagination, so I recommend that no printer filter be resident or that you do a RESET *PR before running this program. You might wish to add line 30 in PRINT/BAS to read: 30 CMD"RESET *PR. The printout never exceeds 64 characters in width, and six filespecs are printed on each page.

The search routine beginning at line 2160 is a sequential search using the INSTR function of LBasic. I could have used a tree search that would be faster, but such a search would require a sorted file. This short routine finds any filespec that contains the sequence of entered characters regardless of where in the filespec that sequence occurs. Line 2420 samples the keyboard during each compare and aborts if you press the up arrow.

The subroutine at line 3720 paginates by checking the current line number stored in the printer device control block, or DCB, and comparing it with 50 to see if a full page has been printed. If so, a form feed goes to the printer and a new header prints across the top of the page. You can select the range of disk numbers to be retrieved; press enter twice to default to print the entire file.

The routines beginning at line 3820 are a copy of the other file-scanning routines, but with the statements in a different order and a somewhat different format. This lets you print out the files by their disk number rather than alphabetically. If you've sorted your database, the files will be alphabetized within each group; otherwise, they'll be in the order in which they were read from the disk's directory.

The error trap in this module (line 5000) is not as complicated as that in the DREADER/BAS module. Its purpose is to return you to the menu if any error should occur. To let you debug, the error code and line number are displayed, and you must press the enter key to return to the menu.

Even though it is less than a full sector in total length, the file TRANSFER/DAT contains information that the program modules cannot function without. If this file does not exist somewhere in the system, all of the modules will assume that you haven't initialized the system yet. Some will abort back to the main menu; others will prompt you to initialize. In any case, if you have any other file named TRANSFER/DAT, the program will read it; if it isn't the right one, pandemonium will break loose. In fact, this file is so important that nothing in the system can be accessed without it.

A disadvantage of the system is that if you buy another drive and wish to change the disk you're storing your data on, you can't do so without modifying the data within this file. You also might be faced with a reconstruct of one of the other files, so I included a utility to deal with these needed changes to TRANSFER/DAT. Since you won't normally be entering this module as part of running this program, this utility has no special menu selection. You type RUN"UTILITY/BAS"; it will prompt you through its operation. You can change any of the system data here that you want to; the changes are not permanent until you exit the program. Know what you are doing and work on a backup of the file! Even though you can abort without making your changes permanent, you can destroy all your files if you aren't careful. When you're finished editing TRANSFER/DAT, the utility will return you to the main menu.

MAGIDEX/CMD

The sole purpose of the file MAGIDEX/CMD is to load and execute LBasic and then load and execute the program MAGIDEX/BAS. You need not key in this part of the program if you don't want to or if you don't know how to use an editor/assembler. It is presented here in the format used by EDAS, and will have to be modified for EDTASM. You'll need to change lines such as 210 into two separate line numbers, each with one byte and one DEFB opcode. Lines to be changed are 180, 210, 220, 230, 240, 250, 260, 270, 280, 290, and 300. Lines 120, 130, and 140 should be deleted since EDTASM does not recognize these pseudo-ops. This loader sets up a temporary buffer at X'B000' for the command LBASIC RUN "MAGIDEX/BAS" and a concluding carriage return. Since no part of this buffer can be clobbered by the loaded program, I have ORGed it in the middle of RAM, well out of harm's way. This loader merely places some graphics and a message on the screen and jumps to X'4405' with the HL register pair pointing to the command to be executed. This vector is in TRSDOS and NEWDOS, and I think in all the other DOSes, and can be used as a way to protect your programs from prying eyes. Here I've used it for convenience, as the programs can be executed from LDOS Ready and will return to command level when finished. Even in this simplified form, such a loader could confuse the new computer user into thinking that your program is tamperproof, especially if you add passwords to all the modules.

If you're using a Model III, you'll have to change a couple of lines. Do not key in SORT50/BAS because there never was an LDOS 5.0 for the Model III. In the module MAGIDEX/BAS, make line 880 read: RUN"SORT/BAS". In the SORT/BAS module, delete lines 160 and 180. In the module PRINT/BAS, you may want to change the up arrow to one of the special characters available on your computer, because a flashing left bracket is not particularly attractive. This should be done in line 1260. The address X'4405' will have to be changed to the correct one for the Model III also. Consult your LDOS manual for the address of the @RUN vector. These should be the only changes required for operation on a Model II under LDOS 5.1.

If you decide to key in these programs, I recommend that you make each statement that contains an ON ERROR GOTO statement into a REM so that the error traps won't throw you when you're debugging your typing. This is good advice for any program, but particularly this one, since an error can cause an immediate return to the menu module and the unrecoverable loss of the program you're trying to debug. After you've corrected all your typing mistakes, you can remove the REM tokens.

Be sure to name the programs exactly as shown, because each module calls the other and they must have the right name. If you're running an LDOS earlier than 5.1.2, you should name the menu module simply MAGIDEX instead of MAGIDEX/BAS to allow for the difference in default extension handling in LDOS 5.1.2.

Disk-drive catalog programs are like spouses: there's only one that's just right for you. This one is exactly right for me, but I've tried to make it easy to modify so that you can make it just right for you. ■

Charles Knight can be reached at 2708 Roberts Circle, Arlington, TX 76010.

Tiger Graphics

by Barbara Clinger

Use the IDS-445 Paper Tiger to print graphics using Basic, Pascal, and Assembly language.

My selection of a printer was based on three things: quality, price, and graphics capabilities. The IDS-445—often called the Paper Tiger—gave me everything I wanted in a printer except lowercase descenders. However, there is a great difference between having a printer with graphics capabilities and actually using it to draw graphs.

The programs in this article illustrate how to use the full graphics capabilities of the Tiger by controlling the individual dots. In fact, these programs graph functions as they would appear on standard graph paper with one-quarter-inch grids.

This article is in two sections; the programs in the first part are written in Level II Disk Basic using NEWDOS80, but will run under TRSDOS 2.3 as well as NEWDOS+. With some modifications they will run with Level II Basic and at least 16K of memory. The

Program Listing 1

```
5 'EXAMPLE 1
10 DEFUSR0=&H7C08:DEFUSR1=&H7D17:DEFUSR2=&H7D65
20 DEFUSR3=&H7C01:CLS
30 DEFINT I,J,T,U,V
40 I6=16:I8=18:P5=.5
50 INPUT"INPUT H";H:POKE&H7DF7,H:POKE&H7DF8,0
60 INPUT"INPUT K";K:POKE&H7DF9,K:POKE&H7DFA,0
70 INPUT"INPUT PHIGH";I:POKE 32239,I:POKE 32240,0:V5=I8*I
80 INPUT"INPUT PWIDE";I:POKE 32241,I:POKE 32242,0
90 Z=USR0(W)
100 DEF FNY(X)=2*X-3
110 INPUT"ENTER THE DOMAIN";A1,A2
120 I1=INT(I6*(A1+H)+P5):I2=INT(I6*(A2+H)+P5)
130 FOR U=I1 TO I2:Z=USR3(U)
140 V=INT(I8*(K+FNY(U/I6-H))+P5)
150 IF V<0 OR V> V5 THEN 210 ELSE Z=USR1(V)
160 IF U=I1 THEN 210
170 IF ABS(T-V)<3 THEN 210
180 IF T>V THEN J1=V+1:J2=T-1
190 IF T<V THEN J1=T+1:J2=V-1
200 FOR I3=J1 TO J2 STEP 2:Z=USR1(I3):NEXT I3
210 T=V
220 NEXT U
225 Z=USR2(W)
230 END
```

The Key Box

Model I
32K RAM
105-445 Printer
Pascal 80
Tiny Pascal

Program Listing 2

```
                 00001          ;TIGERGRAPH
                 00002          ;BARBARA CLINGER
                 00003          ;DEPARTMENT OF MATHEMATICS
                 00004          ;WHEATON COLLEGE
                 00005          ;NORTON, MASS. 02766
                 00006 ;
                 00007 ;
7C01             00010          ORG    7C01H          ;START OF GETU
7C01 CD7F0A      00020 GETU     CALL   0A7FH
7C04 22FB7D      00030          LD     (U),HL
7C07 C9          00040          RET                   ;END OF GETU
7C08             00050          ORG    $              ;START OF AXES
7C08 ED5BEF7D    00060 AXES     LD     DE,(PHIGH)     ;X-Y AXES AT (H0,K0) IN
```

Listing 2 continues

Listing 2 continued

```
7C0C 3E12            LD      A,18             ;    THE PAGE
7C0E 0E00            LD      C,0
7C10 CDDE7D          CALL    MULT             ;PAGEH=18*PHIGH
7C13 22F57D          LD      (PAGEH),HL
7C16 ED5BF17D        LD      DE,(PWIDE)
7C1A 3E10            LD      A,16
7C1C 0E00            LD      C,0
7C1E CDDE7D          CALL    MULT             ;PAGEW=16*PWIDE
7C21 22F37D          LD      (PAGEW),HL
7C24 7D              LD      A,L
7C25 4C              LD      C,H
7C26 ED5BEF7D        LD      DE,(PHIGH)
7C2A CDDE7D          CALL    MULT             ;3*(PAGEW)*(PHIGH) EQUALS
7C2D EB              EX      DE,HL            ;  THE NUMBER OF BYTES IN
7C2E 3E03            LD      A,3                 ;THE PAGE
7C30 0E00            LD      C,0
7C32 CDDE7D          CALL    MULT             ;(HL)=3*(PAGEW)*(PHIGH)
7C35 44              LD      B,H
7C36 4D              LD      C,L
7C37 2100A0    ZERO  LD      HL,PAGE          ;ZERO THE PAGE
7C3A 3600      ZLOOP LD      (HL),0
7C3C 23              INC     HL
7C3D 0B              DEC     BC
7C3E 78              LD      A,B
7C3F B1              OR      C
7C40 20F8            JR      NZ,ZLOOP         ;SEE IF PAGE IS ZEROED
7C42 3E00            LD      A,0
7C44 32FB7D          LD      (U),A            ;U MAKES LEFT HASH MARK
7C47 3E20            LD      A,20H            ;   ON THE Y-AXIS
7C49 32FD7D          LD      (V),A            ;V MAKES THE RIGHT HASH
7C4C 3AF77D          LD      A,(H0)           ;   MARK ON Y-AXIS
7C4F FE00            CP      0                ;SEE IF Y-AXIS = V-AXIS
7C51 CA597C          JP      Z,ISEQU
7C54 3E20            LD      A,20H
7C56 32FB7D          LD      (U),A
7C59 2AF37D    ISEQU LD      HL,(PAGEW)
7C5C 7D              LD      A,L
7C5D 4C              LD      C,H
7C5E 110300          LD      DE,3
7C61 CDDE7D          CALL    MULT             ;FIND 3*PAGEW=NO. SPACES
7C64 E5              PUSH    HL               ;BETWEEN Y MARKS
7C65 3E10            LD      A,16
7C67 0E00            LD      C,0
7C69 ED5BF77D        LD      DE,(H0)
7C6D CDDE7D          CALL    MULT             ;FIND 16*H
7C70 1100A0          LD      DE,PAGE
7C73 19              ADD     HL,DE            ;PAGE+16*H
7C74 E5              PUSH    HL
7C75 2AEF7D          LD      HL,(PHIGH)       ;# OF Y HASH MARKS
7C78 45              LD      B,L
7C79 E1              POP     HL
7C7A D1              POP     DE
7C7B 22037E    MARKY LD      (TEMP),HL
7C7E DD2A037E        LD      IX,(TEMP)        ;IX POINTER TO Y-AXIS
7C82 3AFB7D          LD      A,(U)
7C85 DD77FF          LD      (IX+255),A       ;MARK LEFT OF Y-AXIS
7C88 3AFD7D          LD      A,(V)
7C8B DD7701          LD      (IX+1),A         ;MARK RIGHT OF Y-AXIS
7C8E 19              ADD     HL,DE            ;FOR NEXT HASH MARK
7C8F 10EA            DJNZ    MARKY
7C91 2AF37D          LD      HL,(PAGEW)       ;BEGIN TO MARK X-AXIS
7C94 7D              LD      A,L
7C95 4C              LD      C,H
7C96 ED5BF97D        LD      DE,(K0)
7C9A CDDE7D          CALL    MULT             ;FIND K*PAGEW
7C9D 110300          LD      DE,3
7CA0 7D              LD      A,L
```

Listing 2 continues

Listing 2 continued

```
7CA1 4C                   LD     C,H
7CA2 CDDE7D               CALL   MULT          ;(HL)=PAGE+3*K*(PAGEW)
7CA5 1100A0               LD     DE,PAGE       ;  POINTS X-AXIS
7CA8 19                   ADD    HL,DE
7CA9 2B                   DEC    HL
7CAA E5                   PUSH   HL
7CAB 110F00               LD     DE,15
7CAE A7                   AND    A
7CAF ED52                 SBC    HL,DE
7CB1 111000               LD     DE,16
7CB4 E5                   PUSH   HL            ;(HL)=PAGE+3*K*(PAGEW)-16
7CB5 2AF17D               LD     HL,(PWIDE)    ; LEFTMOST PART OF X-AXIS
7CB8 45                   LD     B,L
7CB9 05                   DEC    B             ;# HASH MARKS ON X-AXIS
7CBA E1                   POP    HL            ;IS (PAGEW)-1
7CBB 7E            MARKX  LD     A,(HL)
7CBC F602                 OR     2             ;MAKE DOT OVER X-AXIS
7CBE 77                   LD     (HL),A
7CBF E5                   PUSH   HL
7CC0 D5                   PUSH   DE
7CC1 ED5BF37D             LD     DE,(PAGEW)
7CC5 19                   ADD    HL,DE
7CC6 22037E               LD     (TEMP),HL
7CC9 DD2A037E             LD     IX,(TEMP)
7CCD DD7E00               LD     A,(IX)
7CD0 F620                 OR     20H           ;MAKE DOT UNDER X-AXIS
7CD2 DD7700               LD     (IX),A
7CD5 D1                   POP    DE
7CD6 E1                   POP    HL
7CD7 A7                   AND    A
7CD8 ED52                 SBC    HL,DE         ;MOVE LEFT FOR NEXT MARK
7CDA 10DF                 DJNZ   MARKX
7CDC 2AEF7D        YAXIS  LD     HL,(PHIGH)
7CDF 7D                   LD     A,L
7CE0 4C                   LD     C,H
7CE1 110300               LD     DE,3
7CE4 CDDE7D               CALL   MULT          ;3*(PHIGH)=# OF BYTES FOR
7CE7 E5                   PUSH   HL            ;   Y-AXIS
7CE8 3E10                 LD     A,16
7CEA 0E00                 LD     C,0
7CEC ED5BF77D             LD     DE,(H0)
7CF0 CDDE7D               CALL   MULT          ;FIND 16*H
7CF3 1100A0               LD     DE,PAGE
7CF6 19                   ADD    HL,DE         ;PAGE+3*(H) POINTS TO
7CF7 D1                   POP    DE            ; START OF Y-AXIS
7CF8 43                   LD     B,E
7CF9 ED5BF37D             LD     DE,(PAGEW)
7CFD 363F          YLOOP  LD     (HL),3FH      ;3FH MAKES COLUMN 6 DOTS
7CFF 19                   ADD    HL,DE
7D00 10FB                 DJNZ   YLOOP
7D02 2AF37D               LD     HL,(PAGEW)    ;START OF X-AXIS
7D05 2B                   DEC    HL
7D06 44                   LD     B,H
7D07 4D                   LD     C,L
7D08 E1                   POP    HL
7D09 7E            XLOOP  LD     A,(HL)
7D0A F601                 OR     1             ;MAKES ONE DOT FOR AXIS
7D0C 77                   LD     (HL),A
7D0D 2B                   DEC    HL            ;MOVE LEFT FOR NEXT DOT
7D0E 0B                   DEC    BC            ;DECREASE COUNTER
7D0F 78                   LD     A,B
7D10 B1                   OR     C
7D11 FE00                 CP     0
7D13 C2097D               JP     NZ,XLOOP      ;SEE IF DONE WITH X-AXIS
7D16 C9                   RET                  ;END OF AXES
7D17                      ORG    $             ;BEGIN OF PLOT
7D17 CD7F0A        PLOT   CALL   0A7FH         ;GET V FROM BASIC
```

Listing 2 continues

```
7D1A 7C                    LD     A,H           ;TO PLOT (U,V)--DIVIDE V
7D1B 4D                    LD     C,L           ;BY 6, GET Q AND R THEN
7D1C 1E06                  LD     E,6           ;SET BIT 5-R IN BYTE
7D1E 1600                  LD     D,0           ;PAGE+Q*(PAGEW)+U
7D20 CDC87D                CALL   DIVIDE
7D23 57                    LD     D,A
7D24 59                    LD     E,C
7D25 ED53FF7D              LD     (QUOT),DE     ;SAVE Q
7D29 22017E                LD     (RMDER),HL    ;SAVE R
7D2C 3AF37D                LD     A,(PAGEW)
7D2F 0E00                  LD     C,0           ;  DIVISOR ON DE
7D31 ED5BFF7D              LD     DE,(QUOT)     ;ON EXIT QUOTIENT IN AD
7D35 CDDE7D                CALL   MULT          ;  REMAINDER IN HL
7D38 ED4BFB7D              LD     BC,(U)
7D3C 09                    ADD    HL,BC
7D3D 0100A0                LD     BC,PAGE
7D40 09                    ADD    HL,BC         ;ADD ADDR OF PAGE
7D41 22037E                LD     (TEMP),HL
7D44 DD2A037E              LD     IX,(TEMP)
7D48 210500                LD     HL,5
7D4B ED5B017E              LD     DE,(RMDER)    ;FIND 5 MINUS RMDER
7D4F A7                    AND    A             ;DETERMINE WHICH BIT
7D50 ED52                  SBC    HL,DE         ;   TO SET
7D52 1601                  LD     D,1
7D54 7D                    LD     A,L
7D55 B7                    OR     A
7D56 2805                  JR     Z,RDONE       ;GO NO MORE ROTATES
7D58 47                    LD     B,A
7D59 CB22           RLOOP  SLA    D             ;ROTATE TO SET BIT
7D5B 10FC                  DJNZ   RLOOP
7D5D DD7E00         RDONE  LD     A,(IX)        ;GET CONTENTS OF BYTE
7D60 B2                    OR     D             ;INSERT NEW INFO IN BYTE
7D61 DD7700                LD     (IX),A        ;STORE BACK INTO BYTE
7D64 C9                    RET                  ;END OF PLOT
7D65                       ORG    $             ;BEGIN OF PRINT
7D65 3E03           PRINT  LD     A,3           ;**
7D67 CDBD7D                CALL   OUTCHR        ;PUT 445 INTO GRAPHICS
7D6A 0E00                  LD     C,0
7D6C ED5BEF7D              LD     DE,(PHIGH)
7D70 CDDE7D                CALL   MULT
7D73 2B             PLOOP  DEC    HL            ;COUNTER VERTICALLY IS
7D74 22FB7D                LD     (U),HL        ;(U)=3*(PHIGH)-1
7D77 EB                    EX     DE,HL
7D78 2AF37D                LD     HL,(PAGEW)
7D7B 7D                    LD     A,L
7D7C 4C                    LD     C,H
7D7D CDDE7D                CALL   MULT          ;(HL)=U*(PAGEW)
7D80 1100A0                LD     DE,PAGE
7D83 19                    ADD    HL,DE         ;(HL)=PAGE+U*(PAGEW)
7D84 110000                LD     DE,0          ;IS THE COUNTER ACROSS
7D87 7E             GETCHR LD     A,(HL)
7D88 FE03                  CP     3             ;**IF CHAR=3,PRINT TWICE
7D8A CCBD7D                CALL   Z,OUTCHR      ;SEND CHARACTER
7D8D CDBD7D                CALL   OUTCHR        ;SEND CHARACTER
7D90 13                    INC    DE            ;INCREASE POINTER
7D91 23                    INC    HL            ;INCREASE POINTER
7D92 E5                    PUSH   HL            ;SAVE POINTER
7D93 2AF37D                LD     HL,(PAGEW)    ;CHECK-SEE IF LINE DONE
7D96 2B                    DEC    HL
7D97 A7                    AND    A
7D98 ED52                  SBC    HL,DE
7D9A 7C                    LD     A,H
7D9B B5                    OR     L
7D9C E1                    POP    HL
7D9D C2877D                JP     NZ,GETCHR     ;CONTINUE IF NOT DONE
7DA0 3E03                  LD     A,3           ;**SEND END OF LINE SGNAL
7DA2 CDBD7D                CALL   OUTCHR
```

Listing 2 continues

Listing 2 continued

```
7DA5 3E0B      02080        LD     A,0BH          ;**
7DA7 CDBD7D    02090        CALL   OUTCHR
7DAA 2AFB7D    02100        LD     HL,(U)         ;CHECK-SEE IF PAGE DONE
7DAD 7C        02110        LD     A,H
7DAE B5        02120        OR     L
7DAF C2737D    02130        JP     NZ,PLOOP       ;CONTINUE IF NOT DONE
7DB2 3E03      02140 DONE   LD     A,3            ;**SIGNAL TO EXIT GRAPHICS
7DB4 CDBD7D    02150        CALL   OUTCHR
7DB7 3E02      02160        LD     A,2            ;**
7DB9 CDBD7D    02170        CALL   OUTCHR
7DBC C9        02180        RET                   ;RET TO CALLING PROG
7DBD F5        02190 OUTCHR PUSH   AF             ;SUBR TO SEND CHAR
7DBE CDD105    02200 CHCKP  CALL   5D1H           ;CHECK STATUS OF PRINTER
7DC1 20FB      02210        JR     NZ,CHCKP       ;WAIT IF NOT READY
7DC3 F1        02220        POP    AF
7DC4 32E837    02230        LD     (37E8H),A      ;SEND IT
7DC7 C9        02240        RET                   ;RET TO PRINT
7DC8 210000    02250 DIVIDE LD     HL,0           ;FROM ZAKS
7DCB 0610      02260        LD     B,16           ;ON ENTRY DIVIDEND IN AC
7DCD CB11      02270 DLOOP  RL     C              ;         DIVISOR IN HL
7DCF 17        02280        RLA                   ;ON EXIT QUOTIENT IN AC
7DD0 ED6A      02290        ADC    HL,HL          ;        REMAINDER IN HL
7DD2 ED52      02300        SBC    HL,DE
7DD4 3001      02310        JR     NC,$+3
7DD6 19        02320        ADD    HL,DE
7DD7 3F        02330        CCF
7DD8 10F3      02340        DJNZ   DLOOP
7DDA CB11      02350        RL     C
7DDC 17        02360        RLA
7DDD C9        02370        RET                   ;END OF DIVIDE SUBROUTINE
7DDE 0610      02380 MULT   LD     B,16           ;FROM ZAKS
7DE0 210000    02390        LD     HL,0           ;ON ENTRY (A)=LSB MULTCND
7DE3 CB39      02400 MLOOP  SRL    C              ;         (C)=MSB MULTCND
7DE5 1F        02410        RRA                   ;         (DE)=MULTIPLIER
7DE6 3001      02420        JR     NC,MLOOP1      ;ON EXIT (HL)=PRODUCT
7DE8 19        02430        ADD    HL,DE
7DE9 EB        02440 MLOOP1 EX     DE,HL
7DEA 29        02450        ADD    HL,HL
7DEB EB        02460        EX     DE,HL
7DEC 10F5      02470        DJNZ   MLOOP
7DEE C9        02480        RET             ;END OF MULT SUBROUTINE
7DEF           02490 PHIGH  EQU    $
0002           02500        DEFS   2
7DF1           02510 PWIDE  EQU    $
0002           02520        DEFS   2
7DF3           02530 PAGEW  EQU    $
0002           02540        DEFS   2
7DF5           02550 PAGEH  EQU    $
0002           02560        DEFS   2
7DF7           02570 H0     EQU    $
0002           02580        DEFS   2
7DF9           02590 K0     EQU    $
0002           02600        DEFS   2
7DFB           02610 U      EQU    $
0002           02620        DEFS   2
7DFD           02630 V      EQU    $
0002           02640        DEFS   2
7DFF           02650 QUOT   EQU    $
0002           02660        DEFS   2
7E01           02670 RMDER  EQU    $
0002           02680        DEFS   2
7E03           02690 TEMP   EQU    $
0002           02700        DEFS   2
A000           02710 PAGE   EQU    0A000H
0000           02720        END
00000 TOTAL ERRORS
```

changes for Level II Basic are given at the end of this section.

The second section introduces new techniques and examples of graphing. However, these applications are developed in Pascal-80 rather than Basic. The Tigergraph program is modified to be used with Pascal-80.

Part I

The graphics dot-plotting mode provides the capability to control through software each print needle during horizontal scans of the print head across a page. The Tiger enters the graphics mode when the control character ASCII ETX (control C or hexadecimal 03) is received. In the graphics mode, the printer uses a raster scan technique allowing a vertical column of seven dots across a

THE MAGIC IS AT YOUR FINGERTIPS.

Dosplus II magic. For you and your business.

The TRS-80 model II was created to help you take care of your business. DOSPLUS II was created to help your TRS-80 help you and your business even more.

DOSPLUS II replaces your O.E.M. DOS with an ultra-sophisticated operating system that is incredibly faster, more reliable and is totally compatible with all standard business oriented software designed for use with the model II. The result is a computer/operating system that far exceeds manufacturer's specs — with more potential and more capacity to be of service to you.

DOSPLUS II operating systems are available for either your floppy or hard disk drives. The hard drive operating system is compatible with Radio Shack, CuCoS, Corvus Micro-Mainframe, Vic Data and other popular hard drive sub-systems.

DOSPLUS II. All the quality and dependability you'd expect from DOSPLUS.

DOSPLUS II DOSPLUS first in quality! First in the industry!

POWERSOFT
A Division of Software Sales, Inc.
11500 Stemmons Fwy., Dallas, Texas 75229
To order call toll free 1-800-527-7432
For product information (214) 484-2976

MICRO SYSTEMS SOFTWARE, INC.
4301-18 Fairlake Circle
Boca Raton, FL 33431
To order call toll free 1-800-327-8724
For product information (305) 983-3390

TRS-80 is a Radio Shack registered trademark of Tandy Corporation, Inc.

page (the seventh dot being repetitious unless printing the bottom row). Bit 7 is not used in the graphics mode and the remaining seven dot positions correspond to bits 0–6 of a memory byte organized as shown in Figs. 1 and 2.

When the printer is in the graphics mode, each control character must be preceded by the control character 03H. For example, to exit graphics mode the characters 03H followed by 02H must be sent to the printer. Consequently, when you want to send 03H as data and not have it be interpreted as a control code, it is necessary to send 03H to the printer twice. The control characters used in this article are 03H to enter the graphics mode, 0BH for vertical tab (end-of-line signal), and 02H to exit graphics.

The initial outline for the program consisted of the step-by-step process I use when I draw a graph. The resulting programs ended up closely following that process. The procedure goes as follows: First take a clean piece of graph paper (I prefer to use graph paper with one-quarter-inch grids.) On this paper draw the coordinate axes and mark off units. Second, plot enough points to determine the shape of the curve. Finally, fill in the sketch using the points as guides.

The three main portions of Tigergraph (Program Listing 2) follow the original outline closely. The Axis routine zeros the page (clean paper), draws the coordinate axes, and marks off the units. The Plot routine does just that; it plots individual points, and the Print routine sends the information to the Tiger. Surprisingly enough, even when plotting points in adjacent spaces, if the graph is rising or falling rapidly, it is necessary to do some vertical filling in of points on the graph.

Now let's discuss the physical properties of the Tiger and define some terms I find useful. Visualize a blank page containing many spaces, each of which must either remain blank or have a dot printed in it. The term "page" will refer to a rectangular array of spaces of a predetermined size. In fact, page is a memory block and contains the information

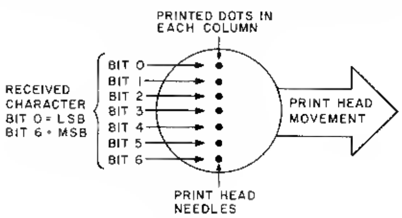


Figure 1

```
5 'POLAR EQUATIONS
6 'CHANGE LINE 1ØØ TO DESIRED POLAR EQUATION
1Ø DEFUSRØ=&H7CØ8:DEFUSR1=&H7D17:DEFUSR2=&H7D65
2Ø DEFUSR3=&H7CØ1:CLS
3Ø DEFINT I,J,T,U,V
4Ø I6=16:I8=18:P5=.5
5Ø INPUT"INPUT H";H:POKE&H7DF7,H:POKE&H7DF8,Ø
6Ø INPUT"INPUT K";K:POKE&H7DF9,K:POKE&H7DFA,Ø
7Ø INPUT"INPUT PHIGH";I:POKE 32239,I:POKE 3224Ø,Ø:V5=I8*I
8Ø INPUT"INPUT PWIDE";I:POKE 32241,I:POKE 32242,Ø
9Ø Z=USRØ(W)
1ØØ DEF FN R(A)=2*(1 - COS(A))
13Ø FOR A=Ø TO 6.2 STEP .Ø5
14Ø U=INT(I6*(H+FN R(A)*COS(A))+P5):Z=USR3(U)
145 V=INT(I8*(K+FN R(A)*SIN(A))+P5):Z=USR1(V)
22Ø NEXT A
225 Z=USR2(W)
```

Program Listing 3

Program Listing 4

```
1Ø 'PROGRAM CONICS/BAS
15 'BARBARA CLINGER
2Ø 'DEPARTMENT OF MATHEMATICS
25 'WHEATON COLLEGE
3Ø 'NORTON, MASS.  Ø2766
4Ø DEFUSRØ=&H7CØ8:DEFUSR1=&H7D17:DEFUSR2=&H7D65
45 DEFUSR3=&H7CØ1
5Ø DEFINT I,J,T,U,V
55 I6=16:I8=18:P5=.5
57 PRINT"PHIGH = PAGE HEIGHT   PWIDE = PAGE WIDTH
6Ø INPUT"INPUT PHIGH";I:POKE 32239,I:POKE 3224Ø,Ø:V5=I8*I
65 INPUT"INPUT PWIDE";I:POKE 32241,I:POKE 32242,Ø
7Ø INPUT"INPUT HØ";H:POKE&H7DF7,H:POKE&H7DF8,Ø
75 INPUT"INPUT KØ";K:POKE&H7DF9,K:POKE&H7DFA,Ø
8Ø Z=USRØ(W)
85 INPUT"CHOICES (1) INPUT CONIC  (2) DRAW (3) END";I
9Ø ON I GOTO 1ØØ,95,725
95 Z=USR2(W):GOTO 85  'ZERO PAGE AND DRAW AXES
1ØØ PRINT"CHOICES OF CONICS TO GRAPH:";TAB(3Ø)"(1) LINE
1Ø5 PRINT TAB(3Ø);"(2) PARABOLA
11Ø PRINT TAB(3Ø);"(3) CIRCLE
115 PRINT TAB(3Ø);"(4) ELLIPSE
12Ø PRINT TAB(3Ø);"(5) HYPERBOLA
125 INPUT"ENTER THE APPROPRIATE NUMBER ";J
13Ø IF J<1 OR J>5 THEN 1ØØ
135 ON J GOTO 15Ø,32Ø,45Ø,45Ø,565
145 'DRAW LINE
15Ø PRINT"OPTIONS FOR FORM OF LINE:";TAB(3Ø)"(1) TWO-POINT
155 PRINT TAB(3Ø);"(2) POINT-SLOPE
16Ø INPUT"   ENTER THE APPROPRIATE NUMBER";I
165 IF I<1 OR I>2 THEN 15Ø
17Ø INPUT"ENTER X1 AND Y1";X1,Y1
175 IF I=1 THEN 195
18Ø INPUT"ENTER THE SLOPE";M
185 X2=X1+1:Y2=Y1+M
19Ø GOTO 2Ø5
195 INPUT"ENTER X2 AND Y2";X2,Y2
2ØØ IF X1=X2 THEN 27Ø
2Ø5 IF X1<=X2 THEN 215 ELSE X3=X1:X1=X2:X2=X3
21Ø Y3=Y1:Y1=Y2:Y2=Y3
215 U1=I6*(X1+H):V1=I8*(Y1+K)
22Ø U2=I6*(X2+H):V2=I8*(Y2+K)
225 M=(V2-V1)/(U2-U1):B=(U2*V1-U1*V2)/(U2-U1)
23Ø INPUT "ENTER DOMAIN";A1,A2
235 I1=INT(I6*(A1+H)+P5):I2=INT(I6*(A2+H)+P5)
24Ø FOR I=I1 TO I2
245 Z=USR3(I)
25Ø V=INT(M*I+B+P5)
255 GOSUB 685
26Ø NEXT I
265 GOTO 85
27Ø INPUT"LINE IS PARALLEL TO Y-AXIS, INPUT RANGE OF Y'S";Y1,Y2
275 J1=INT(I8*(Y1+K)+P5):J2=INT(I8*(Y2+K)+P5)
28Ø U=INT(I6*(X1+H)+P5):Z=USR3(U)
285 FOR I3=J1 TO J2
29Ø IF I3>V5 OR I3<Ø THEN 3ØØ
295 Z=USR1(I3)
3ØØ NEXT I3
3Ø5 GOTO 85
315 'GRAPH PARABOLA
32Ø PRINT"TWO OPTIONS: (1) OPENS UP/DOWN (2) OPENS LEFT/RIGHT"
325 PRINT"(1)  (X-H1)[2=M(Y-K1)  (2)  (Y-K1)[2=M(X-H1)
```

Listing 4 continues

for the axes and the graph.

The vertical spacing of dots is constant for all characters per inch and is fixed at .014 inch per dot. The horizontal spacing must be chosen to give spacing as close to .014 as possible. The print density of a line (characters per inch) controls the horizontal dot spacing, the options being 8.3, 10, and 12 cpi (16 cpi is not recommended for graphics). At 12 cpi the spacing is .0156 inch per dot and is the option nearest to .014 inch. The vertical spacing of .014 inch per dot gives 72 dots per inch. Thus, 18 vertical dots give a unit of one-quarter inch. At 12 characters per inch, the horizontal spacing is .0156 per dot (64.2 dots per horizontal inch) giving a 16-dot-per-one-quarter-inch unit. The error this scale produces is minimal; for example, if a page is to be a square with 6 inches on each side, the difference in the dimensions is only .0576 of an inch—certainly not obvious to the human eye. Thus, using 12 cpi is satisfactory to produce the desired units.

In order to simplify the placement of a dot within a space in a page, I have introduced a set of u-v coordinates where the units are horizontal and vertical spaces. This system is only for reference and is never printed. The origin of the u-v system is at the bottom left corner of the page.

In addition to referring to rows of spaces or columns of spaces, the term "line" is introduced. Since each pass of the print head controls six vertical dots, the term line is defined to be six rows of spaces, i.e., six spaces high by the number of columns in a page. In fact, a line corresponds to one pass of the print head. The problem of placing a dot in a page is simply one of selecting the line and column in which a dot is located, then setting the appropriate bit.

In Fig. 3 the page is two inches square and has eight units in each direction. Thus there are $8 \times 16 = 128$ columns in the u direction, $8 \times 18 = 144$ rows in the

Listing 4 continued

```
330 INPUT"ENTER APPROPRIATE NUMBER";J
335 INPUT"ENTER THE CONSTANTS H1, K1, AND M";H1,K1,M
340 INPUT"ENTER THE DOMAIN ";I1,I2
345 I1=INT(I6*(I1+H)+P5):I2=INT(I6*(I2+H)+P5)
350 IF J=2 THEN 385
355 A=9/(128*M)
360 FOR I=I1 TO I2:Z=USR3(I)
365 V=INT(I8*(K+K1)+A*(I-I6*(H+H1))[2+P5)
370 GOSUB 685
375 NEXT I
380 GOTO 85
385 IF M>0 AND I1<I6*(H+H1) THEN I1=I6*(H+H1)
390 IF M<0 AND I2>I6*(H+H1) THEN I2=I6*(H+H1)
395 FOR I=I1 TO I2:Z=USR3(I)
400 V=INT(I8*(K+K1)+(I8/SQR(I6))*SQR(M*(I-16*(H+H1)))+P5)
405 GOSUB 685
410 NEXT I
415 FOR I=I1 TO I2:Z=USR3(I)
420 V=INT(I8*(K+K1)-(I8/SQR(I6))*SQR(M*(I-I6*(H+H1)))+P5)
425 GOSUB 685
430 NEXT I
435 GOTO 85
445 'GRAPH CIRCLE OR ELLIPSE
450 INPUT"ENTER THE COORDINATES OF THE CENTER";H1,K1
455 U=INT(I6*(H+H1)+P5):V=INT(I8*(K+K1)+P5)
460 Z=USR3(U):Z=USR1(V)   'PLOT THE CENTER
465 IF J=3 THEN INPUT"ENTER THE RADIUS SQUARED";A:B=A:GOTO475
470 INPUT"ENTER A AND B SQUARED";A,B
475 INPUT"ENTER THE DOMAIN";A1,A2
480 IF A1<H1-SQR(A) THEN A1=H1-SQR(A)
485 IF A2>H1+SQR(A) THEN A2=H1+SQR(A)
490 I1=INT(I6*(H+A1)+P5):I2=INT(I6*(H+A2)+P5)
495 M=(9*SQR(B))/(8*SQR(A)):M1=I6[2*A:M2=I6*(H+H1)
500 FORI=I1 TO I2:Z=USR3(I)
505 IF M1<(I-M2)[2 THEN V=INT(I8*(K+K1)+P5):GOTO 515
510 V=INT(I8*(K+K1)-M*SQR(M1-(I-M2)[2)+P5)
511 IF I=I1 THEN T1=V
512 IF I=I2 THEN T2=V
515 GOSUB 685
520 NEXT I
525 FOR I=I1 TO I2:Z=USR3(I)
530 IF M1<(I-M2)[2 THEN V=INT(I8*(K+K1)+P5):GOTO 540
535 V=INT(I8*(K+K1)+M*SQR(M1-(I-M2)[2)+P5)
536 IF I=I1 THEN T3=V
537 IF I=I2 THEN T4=V
540 GOSUB 685
545 NEXT I
546 I=I1+1:T=T2:V=T4:GOSUB 685
547 Z=USR3(I1):T=T1:V=T3:GOSUB 685
550 GOTO 85
555 'GRAPH'GRAPH HYPERBOLA
565 PRINT"TWO OPTIONS: (1) REAL AXIS PARALLEL TO X-AXIS
570 PRINT"              (2) REAL AXIS PARALLEL TO Y-AXIS
575 INPUT"INPUT APPROPRIATE INTEGER";J
580 INPUT"ENTER THE CONSTANTS H1 AND K1";H1,K1
585 INPUT"INPUT A AND B SQUARED";A,B
590 INPUT"ENTER DOMAIN";A1,A2
595 M=((-1)[(J)*I6[2*A:M1=(I8*SQR(B))/(I6*SQR(A))
600 M2=I6*(H+H1):M3=INT(I6*(H+H1-SQR(A))+P5):M4=INT(I6*(H+H1+SQR(A))+P5)
605 I1=INT(I6*(H+A1)+P5):I2=INT(I6*(H+A2)+P5)-1
610 FOR I=I1 TO I2
615 IFJ=1 AND I>M3 AND I<M4 THEN I=M4:T=INT(I8*(K+K1)+M1*SQR((I-M2)[2+M)+P5)
620 Z=USR3(I)
625 V=INT(I8*(K+K1)+M1*SQR((I-M2)[2+M)+P5)
630 GOSUB 685
635 NEXT I
637 IF I=I2 THEN T4=V
640 FOR I=I1 TO I2
645 IFJ=1 AND I>M3 AND I<M4 THEN I=M4:T=INT(I8*(K+K1)-M1*SQR((I-M2)[2+M)+P5)
650 Z=USR3(I)
655 V=INT(I8*(K+K1)-M1*SQR((I-M2)[2+M)+P5)
660 GOSUB 685
665 NEXT I
670 GOTO 85
680 'SUBROUTINE TO FILL, IF NECESSARY
685 IF V<0 OR V>V5 THEN 715 ELSE Z=USR1(V)
690 IF I=I1 THEN 715
695 IF ABS(T-V)<3 THEN 715
700 IF T>V THEN J1=V+1:J2=T-1
705 IF T<V THEN J1=T+1:J2=V-1
710 FOR I3=J1 TO J2 STEP 2:Z=USR1(I3):NEXT I3
715 T=V
720 RETURN
725 END
```

v direction, and $144/6 = 24$ lines—each line containing 128 bytes. The dot A in Fig. 3 that is in the 18th column, 10th row of the page has for its u-v coordinates (18, 10). You must set bit 1 in byte 146 and send it to the printer to print this dot. When an entire page is sent to the printer, it will be sent from the top line down to line 0.

To compute the memory requirements for a page, multiply the number of columns by the number of lines. In a 2-by-2 page, the memory requirement is $128 \times 24 = 3{,}072$ (C00 in hexadecimal). In the case of a 16K machine, the maximum size page is approximately 2-by-2 square inches with eight units in each direction. Another limitation on the page size results from the POKE statements in Basic—the second argument (the value) must be an integer and must be less than 256. For the program Tigergraph as listed, the dimensions of the page should be no more than 15 units wide by 14 units high since $15 \times 16 = 240$ is less than 256 and $14 \times 18 = 252$ is also less than 256.

In most cases these maximum dimensions are adequate since they give a page measuring 3.75 inches wide by 3.5 inches high. However, in Tigergraph all variables are double-precision integers; so by POKEing both the least-significant and most-significant bytes of the coordinates of a point into memory, you can use a larger page. In Tigergraph the starting location of page is A000H (40,960 in decimal) and the origin is 7C01H (31,745 in decimal). For a machine with 16K memory, use 7000H (28,672 in decimal) for page and 7C01H for the origin.

Using the definitions above, the crux of the problem is to make specific dots print. Not printing in a space is simple—send a zero for that bit. Printing a dot in a given space uses two simple concepts: integer division and the greatest-integer function.

In elementary school, the first type of division you encounter is integer division; an integer A divided by a nonzero integer B gives a quotient Q and a remainder R, where R is equal to or greater than zero and less than B. For example, 10 divided by 6 gives a quotient 1 and a remainder of 4. Notice that the quotient is given by the greatest-integer function, denoted in Basic by INT. That is, INT(10/6) equals INT(1.666...) equals 1.

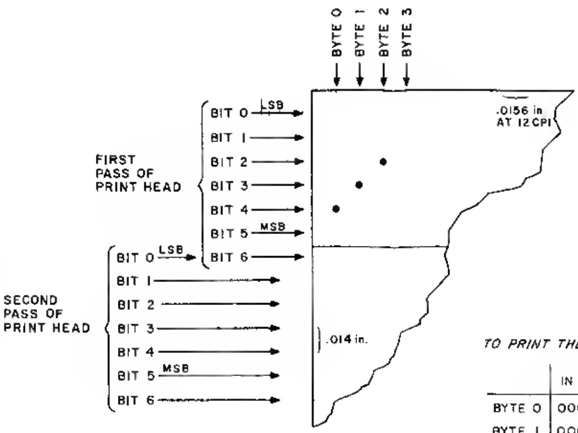
By definition, the greatest-integer function (INT) is defined to be the largest integer not exceeding the argument of the function:

INT(2.49) = 2　　INT(3.001) = 3
INT(2.999) = 2　　INT(3) = 3

A second property of the Basic INT function is: Given a real number x, the integer nearest x is found by INT(x + .5). For example, the integer nearest 2.49 is 2, and INT(2.49 + .5) equals INT(2.99) equals 2. The integer nearest 2.51 is 3, and INT(2.51 + .5) equals INT(3.01) equals 3.

More information on the greatest-integer function and its properties can be found in most number-theory texts.

Two examples illustrate the procedure by which dots can be printed. Continuing with the example of a 2-by-2 page, the points A and B marked on Fig. 3 correspond to Examples 1 and 2, respectively.



TO PRINT THE THREE DOTS:

	IN BINARY	IN HEXADECIMAL	IN DECIMAL
BYTE 0	00010000	10	16
BYTE 1	00001000	08	8
BYTE 2	00000100	04	4

Figure 2

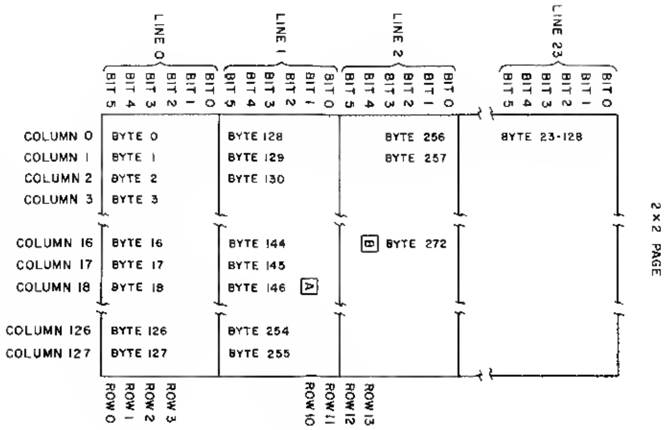


Figure 3

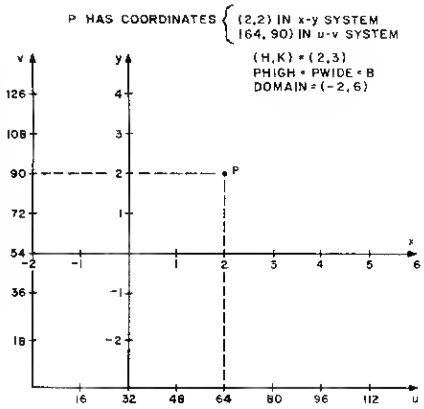
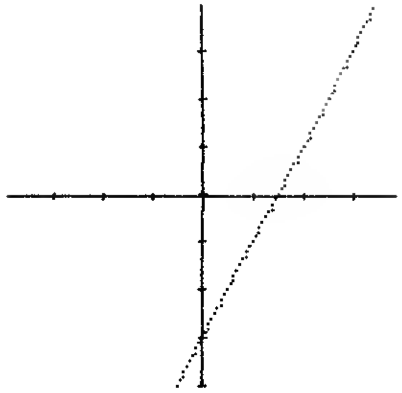


Figure 4



INPUT H? 4
INPUT K? 4
INPUT PHIGH? 8
INPUT PWIDE? 8
ENTER THE DOMAIN? −4,4

Figure 5

"LEARNING IS CHILD'S PLAY WITH EDUGAMES® BY CLS"

WORDFLASH® WORDMAZE®

Ageless Teaching Concepts Brought Into The Computer Era

INNOVATIVE TECHNIQUES ADD EXCITEMENT AND CHALLENGE!

- Fast moving, high scoring, instant execution holds interest
- All options user-controlled for maximum "hands-on" experience
- Multi-list capability; create own lists; add, delete, change at will; basic word list included with each program
- Timing/recordkeeping features challenge user to upgrade, allow for competition
- Choice of lists; grade levels; language difficulty; print out; speed and more
- Perfect for all the family; customize to individual needs
- Mathflash option; all arithmetic functions by curriculum objective

FOR HOME OR SCHOOL, CREATE LARGE VARIETY OF EXCITING, INTERESTING, CHALLENGING LESSON UNITS IN GAME OR PRACTICE FORM

All EDUGAMES® fully tested at Center for Unique Learners

• Field tested (grade 1 - college) • Debugged by teacher/programmers • Related to teaching objectives

The Center for Unique Learners is a non-profit teaching institution working with youngsters and adults at every grade level It is directed by Dr Nancy Dworkin, noted expert, author, lecturer, international consultant on reading and teaching systems

WORDFLASH® — beginning to complex language manipulation

- Practice spelling, recognition, grammar; by words or sentences
- Alphabet or random order; control number of attempts and timing, size of letters
- Final printout cross tabulates words used, errors, error patterns, number of attempts each word, percent correct first try, percent correct total, raw score, modified score *(disk only)*

Cassette Version $24 95
Disk Version (includes extensive record keeping function and other options) $49.95

WORDMAZE® — endless puzzles in random array

- Choice of puzzle width, length, number of words; printouts by identical puzzles or different with same list
- Answer key on screen/print
- High Score "Hall of Fame" constantly updated *(disk only)*

Cassette Version $19 95
Disk Version $39 95

ARCADE VERSIONS

I. Above plus option for multiple players competing using same words with different puzzle matrix, individual running score maintained
Disk Only $49 95

II Both above plus option for two players, same puzzle at same time! True, arcade challenge—great excitement
Disk Only $59 95

MATHFLASH®

- Add, subtract, multiply, divide
- Full control options as above plus many special math-related options such as decimal, carrying, borrowing

As option with either of above $10 00
MATHFLASH® alone
Cassette Version $24 95
Disk Version $39 95

WORDFLASH®/WORDMAZE®/MATHFLASH®:

Combination system, all 3 programs $85.00
Manual only, full educational documentation, suggested uses, etc (cost credited) $10 00
Expanded multi-teacher/multi-student recording school system available ...... inquire

CLS Associatee
A TEACHING EXTRAVAGANZA

CLS Associates

Computer Learning Systems

P.O. BOX 6147, SILVER SPRING, MARYLAND 20906, (301) 924-2551

✓376

ORDER FORM

Please send

- ☐ WORDFLASH™
  - ☐ Cassette $19 95
  - ☐ Disk $49 95
- ☐ WORDMAZE™
  - ☐ Cassette $19 95
  - ☐ Disk $39 95
  - ☐ ARCADE I (Disk Only) $49.95
  - ☐ ARCADE II (Disk Only) $59 95
- ☐ MATHFLASH™
  - ☐ Add to any above $10 00
  - MATHFLASH™ alone
  - ☐ Cassette $19 95
  - ☐ Disk $39 95
- ☐ Combination system, all 3 ...... $85 00
  - Manual alone
  - ☐ WORDFLASH™ $10 00
  - ☐ WORDMAZE™ $10 00

Check equipment

- ☐ TRS-80
  - ☐ Model I
  - ☐ Model III
  - ☐ Single disk
  - ☐ Two disk
- ☐ APPLE II+

TRS-80 is a trademark of Radio Shack
APPLE is the trademark of APPLE Corporation

☐ ADD $2 00 SHIPPING IN U.S A
☐ ADD $4.00 C.O D
☐ MASTERCHARGE ☐VISA

Card # ____________

Exp Date ____________

Name ____________

Address ____________

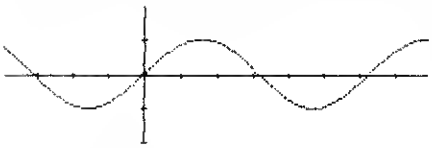
City ____________

State ____________ Zip ____________

✓See List of Advertisers on Page 563

Example 1

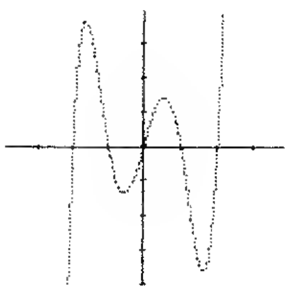
Suppose the dot A in the page is 18 spaces in the u direction and 10 spaces up, i.e., A(u,v) = (18,10). A is in line 1 and column 18; so, you want to set bit 1 of byte number 146. In fact, the quotient (found by the greatest-integer function) gives you the means to determine which byte a given point is in. In this example, divide 10 by 6 getting the quotient Q = 1 and R = 4. (Six is used as a divisor because there are six rows per line.) The quotient 1 gives the line in which the byte is located, and five minus the remainder (5 − 4 = 1) is the bit we wish to set. In order to set bit 1 of that byte, we store 00000010 in binary (2 in decimal) into it. Since each line of the page consists of 128 bytes, the byte in line 1, column 18 is 1 × 128 + 18 = 146th byte into the page.



INPUT H? 4
INPUT K? 2
INPUT PHIGH? 4
INPUT PWIDE? 12
ENTER THE DOMAIN? −4,8

Y = SIN(X)

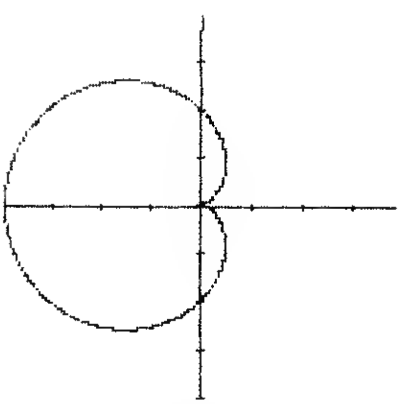
Figure 6



INPUT H? 4
INPUT K? 4
INPUT PHIGH? 8
INPUT PWIDE? 8
ENTER THE DOMAIN? −4,4

Y = X∗(X − 2)∗(X − 1)∗(X + 1)∗(X + 2)

Figure 7



INPUT H? 4
INPUT K? 4
INPUT PHIGH? 8
INPUT PWIDE? 8

Figure 8

Example 2

The point B(u,v) = (16,13) is in line 2, column 16. You must insert a 1 in bit 4 of byte 272. Dividing 13 by 6, you find the quotient is 2 and the remainder is 1. Thus, the point B (in line 2) is in byte 2 × 128 + 16 = 272 and the bit to be set is 5 − 1 = 4.

In both examples the bytes would be offset by the starting address of the page. For example, if the starting address of page is 7000H, the byte of Example 2 is 28672 + 2 × 128 + 16.

In general, to plot an arbitrary point (A,B), proceed as follows: Divide B by 6; get the quotient Q and the remainder R. Insert 1 into bit (5-R) of the byte in the page given by (PAGE) + Qx (PAGEW) + A, where (PAGEW) is the number of columns in page and (PAGE) is the starting address in memory of the page.

Unfortunately, graphs in the u-v coordinate system are too small to be useful. Besides, we think of points in terms of an x-y system. Beginning with a point in the x-y system necessitates a change of coordinates to the u-v system in order to plot the specific point. Before going into more detail on the transformation of coordinates—found in most texts in analytic geometry or introduction to calculus—an example is helpful. Continuing with the square 2-by-2 page, suppose I want the x-y coordinate axes to be two quarter-inch units to the right of the v axis and three quarter-inch units above the u axis. (See Fig. 4.)

Since each vertical unit is 18 spaces, the x axis is 3 × 18 = 54 spaces up from the bottom of the page. Similarly, the y axis is 2 × 16 = 32 spaces from the left edge of the page. To plot the point P, whose coordinates are (2,2) in the x-y coordinate system, you see that the point is a total of four units or 4 × 16 = 64 spaces in the u direction and is five units or 5 × 18 = 90 spaces in the v direction. Hence, to plot the point (2,2) in the x-y system, you actually plot the point (64,90) in the u-v system.

In changing from the x-y coordinate system to the second coordinate system (the u-v coordinate system), with the origin of the second system at the point (−H, −K) (H and K are positive integers given in terms of the x-y coordinates), the change of coordinates would be u = x + H and v = y + K.

However, not only are you translating the axes, but you also want to shrink the units from quarter-inch units horizontally and vertically to one space in each direction in the u-v system. That transformation of coordinates is given by u = 16(x + H) and v = 18(y + K).

The following convention is followed throughout the programs in this article:

The integers H and K for the placement of the x-y coordinate system into a page are positive integers. H equals the number of units the y axis is to the right of the v axis and K equals the number of units the x axis is above the u axis. Thus, the transformation of coordinates are: First, to change from the u-v system to the x-y system, u = 16(x + H) and v = 18(y + K). Second, to change from the x-y system to the u-v system: x = (u/16) − H and y = (v/18) − K. (Don't panic—the programs do this for you.)

Once the coordinate system has been changed, it is also necessary to change the variables in the defining relation of the function. That is, if I graph the function y = f(x) in the x-y system, then the equivalent function in the u-v system will be (v/18) − K = f((u/16) − H), or equivalently v = 18(K + f(u/16 − H)).

For example, for (H,K) = (2,3) and for the function given by the equation y = 3x + sin(x), substituting x = u/16 − 2 and y = v/18 − 3 gives v/18 − 3 = 3(u/16 − 2) + sin(u/16 − 2).

Simplifying the last equation gives v = 18(3 + 3(u/16 − 2) + sin(u/16 − 2)).

The values of u and v are calculated in a Basic program and sent to the Plot routine in Tigergraph. The values for u generated in the Basic program are generated as integers, but the values of v computed using the formula above usually are not. Since u and v represent spaces, you must send the nearest integer value for v. As pointed out above, this is accomplished by using the INT function: v = INT(18∗(K + f(u/16 − H)) + .5), or in general, v = INT(v + .5).

For example, Program Listing 1 (EXAMPLE1/BAS) is a Basic program that uses Tigergraph to draw the line y = 2x − 3. The equation of this function is defined in line 100, and values of v are computed in line 140.

Prior to running EXAMPLE1/BAS, load TIGER/OBJ, and then call Basic. The memory size should be set to 31,743, then when the program is run, six numbers must be supplied in response to the input statements in lines 50, 60, 70, 80, and 110. The values for

H, K, PHIGH, and PWIDE must be positive integers, but the values for the domain can be real numbers. (H and K are defined above, PHIGH, PWIDE, and the domain are defined in the next paragraph.) A list of the responses and the resulting graph are shown in Fig. 5.

Here is a brief description of the Basic program. Lines 10 and 20 give the starting addresses of the four routines of Tigergraph. The first routine, called by USR0, inserts zeros into the page and draws the x and y coordinate axes with hash marks at quarter-inch intervals. USR3 and USR1 send the appropriate values of u and v, respectively, into the Plot routine. USR2 sends the page to the Tiger. Line 20 also clears the screen.

Line 30 defines all variables with names beginning with I, J, T, U, and V to be integer variables. Line 40 defines the variables I6, I8, and P5 to be 16, 18, and .5, respectively. Using variable names, the program runs faster than it would if the specific values were used in the computational portions of the program. (See the section on speeding up execution in the *Level II Basic Reference Manual.*)

Lines 50 and 60 ask for the coordinates (H,K), which place the x-y axes as described above. In the Tigergraph listing and CONICS/BAS, H and K are represented by H0 and K0.

Lines 70 and 80 ask for values that define the length of the page in memory. In particular, PHIGH is the number of units for the y axis, each unit being one-quarter inch. Thus, the answer of eight units makes the y axis 2 inches high. PWIDE is the number of units for the x axis, and again an answer of eight gives an x axis 2 inches wide. In Tigergraph these numbers are double-precision integers, so the values are POKEd into the least-significant byte and zero is POKEd into the most-significant byte.

Line 90 calls USR0, which zeros the PAGE and draws the axes. Line 100 defines the function to be graphed.

Line 110 asks for the domain of the function. That is, it asks for the range of the x value for which the graph is to be drawn. These values must be equal to or greater than −H and less than or equal to PWIDE − H.

Line 120 converts the domain of the function in terms of x to the equivalent domain in terms of u. These values for the domain of u are then used in line 130, u going from the lower limit to the upper limit of the domain. Also in this line, USR3 sends the current value of u to the Plot routine of Tigergraph. The function must be defined for all x values in its domain.

Line 140 computes the corresponding values for v. Line 150 tests the value of v to be sure that it lies within the upper and lower boundaries of the page. Specifically, v must be nonnegative and less than or equal to the number of rows in the page, which is 18 times (PHIGH). Only if the value of v is within these limits is it sent into Plot.

Lines 160–210 test for the vertical separation between consecutive v values, and if they are too widely spaced, points are plotted to fill in the gap. Specifically, line 160 tests to see if the current v value is the first point or not. If it is the first point, no filling is necessary. Otherwise, when the last computed value of v (stored in variable T) is more than two spaces from the current v value, the gap is filled in. Lines 180 and 190 determine whether the fill goes up or down, and line 200 does the filling. Line 210 saves the current v value in T to be the previous value next time through this loop.

Line 220 ends the loop begun in line 130. Line 225 uses the Tigergraph's Print routine to print the page.

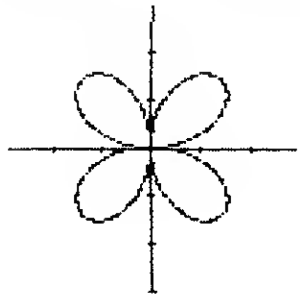
This program can be run for functions that are defined for all x's in the domain and, at least piecewise, continuous—that is, continuous at all but a finite number of points. All functions must give y explicitly as a function of x. By definition, a function is single valued. In order to graph other functions, simply change the definition in line 100. (See Figs. 6 and 7.) It is possible to graph several functions on the same graph. In order to do this, run the Basic program as it is for the first function. Next, edit the Basic program by deleting line 90; otherwise, the first graph will be erased. Then change line 100 to define the new function and run the program again.

Tigergraph has four main routines, labeled in the listing as Getu, Axes, Plot, and Print. The role of each routine has been discussed above. The variables within the program are PHIGH, the number of units for the y axis; PWIDE, the number of units for the x axis; PAGEW, the number of spaces for the u axis (16 × PWIDE); PAGEH, the number of spaces for the v axis (18 × PHIGH); H0 and K0, the placement of the x and y axes in the page; U and V, the point to be placed into the page; and PAGE, the starting memory address for the page.

Other routines in Tigergraph are for computational purposes. The subroutines for double-precision addition and multiplication are from *How to Program the Z80* by Rodnay Zaks.

Now that the specifics of the programming techniques are completed, there are two Basic programs that I find useful. The first is POLAR/BAS (Program Listing 3) and the second is CONICS/BAS (Program Listing 4).

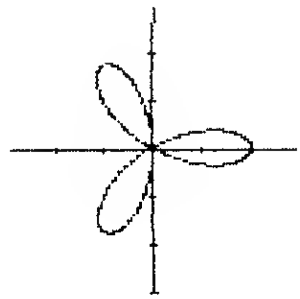
A particularly tedious type of function to graph is one that is given in terms



INPUT H? 3
INPUT K? 3
INPUT PHIGH? 6
INPUT PWIDE? 6

R = 2*SIN(2*A)

Figure 9



INPUT H? 3
INPUT K? 3
INPUT PHIGH? 6
INPUT PWIDE? 6

R = 2*COS(3*A)

Figure 10

Standard Form of Conics

(1) Parabola: vertex at (h,k)

(a) $(x-h)^2 = M(y-k)$ opens up if M>0, opens down if M<0

(b) $(y-k)^2 = M(x-h)$ opens right if M>0, opens left if M<0

(2) Circle: Center at (h,k), radius r

$(x-h)^2 + (y-k)^2 = r^2$

(3) Ellipse: center at (h,k)

$$\frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1$$

major axis parallel to x axis if a>b
major axis parallel to y axis if a<b

(4) Hyperbola: center at (h,k)

(a) $\dfrac{(x-h)^2}{a^2} - \dfrac{(y-k)^2}{b^2} = 1$

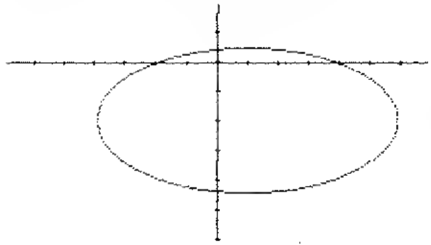
real axis parallel to x axis

(b) $\dfrac{-(x-h)^2}{a^2} + \dfrac{(y-k)^2}{b^2} = 1$

real axis parallel to y axis

Figure 11

of polar coordinates. The program POLAR/BAS graphs the equation $R=2(1-\cos(A))$, and the resulting graph is given in Fig. 8. In POLAR/BAS, the polar equation is defined in line 100. Lines 140 and 145, in addition to changing to the u-v coordinate system, also change from polar coordinates to rectangular coordinates. Recall that this change of coordinates is given by



```
>RUN
INPUT H? 7
INPUT K? 6
INPUT PHIGH? 8
INPUT PWIDE? 14
CHOICES (1) INPUT CONIC
  (2) DRAW (3) END? 1
CHOICES OF CONICS TO GRAPH:
  (1) LINE
  (2) PARABOLA
  (3) CIRCLE
  (4) ELLIPSE
  (5) HYPERBOLA
ENTER THE APPROPRIATE NUMBER? 4
ENTER THE COORDINATES OF THE
  CENTER? 1,-2
ENTER A AND B SQUARED? 24,6
ENTER THE DOMAIN? -7,7
```

Figure 12

$$x = r\cos A$$
$$y = r\sin A$$

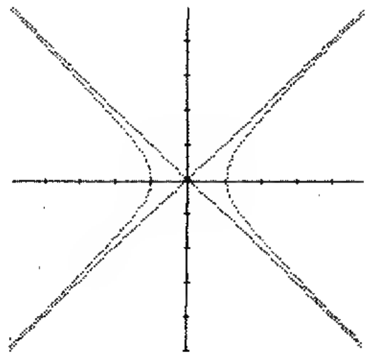
where A is the angle (in radians) going from the polar axis to the line segment joining (x,y) and the origin. Therefore, lines 140 and 145 result from the following: First, changing from polar to rectangular coordinates gives

$$x = 2(1-\cos A)\cos A$$
$$y = 2(1-\cos A)\sin A$$

Then changing to the u-v coordinates using $x=u/16-H$ and $y=v/18-K$ gives

$$\frac{u}{16} - H = 2(1-\cos A)\cos A$$

$$\frac{v}{18} - K = 2(1-\cos A)\sin A$$



HYPERBOLA $x^2 - y^2 = 1$
ASYMPTOTES $y = \pm x$

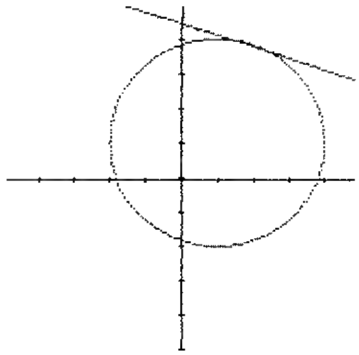
Figure 13

and this simplifies to

$$u = 16[H + 2(1-\cos A)\cos A]$$
$$v = 18[K + 2(1-\cos A)\sin A]$$

The final program (CONICS/BAS) draws the various conic sections. This program lets you graph any combination of lines, parabolas, circles, ellipses, and hyperbolas using one set of coordinate axes. To use this program, it is necessary to have the equation of a conic in its standard form. (See Fig. 11 for a summary of the standard forms of the conics.) Since the program is rather long, I have kept its prompts as brief as possible.

The program to line 80 is identical to the corresponding portions in Program Listing 1, and the discussion of CON-



CIRCLE $(x-1)^2+(y-1)^2=9$
TANGENT LINE AT (2,3,83)

Figure 14

Program Listing 5

```
                00001          ;TIGERGRAPH
                00002          ;BARBARA CLINGER
                00003          ;DEPARTMENT OF MATHEMATICS
                00004          ;WHEATON COLLEGE
                00005          ;NORTON, MASS. 02766
                00006 ;
                00007 ;
7C01            00010          ORG     7C01H
7C01 CD7F0A     00012          CALL    0A7FH           ;GET NUMBER
7C04 7D         00013          LD      A,L
7C05 FE00       00014          CP      0               ;COMPARE IT TO ZERO
7C07 C20E7C     00016          JP      NZ,NOTZ         ;JUMP IF NOT ZERO
7C0A CD1C7C     00018          CALL    AXES
7C0D C9         00020          RET
7C0E FE01       00022 NOTZ     CP      1               ;COMPARE IT TO ONE
7C10 C2177C     00024          JP      NZ,NOTO         ;JUMP IF NOT ONE
7C13 CD2C7D     00026          CALL    PLOT
7C16 C9         00028          RET
7C17 CD7B7D     00030 NOTO     CALL    PRINT           ;MUST BE 3
7C1A C9         00032          RET
7C1B 00         00050          NOP                     ;START OF AXES
7C1C ED5B057E   00060 AXES     LD      DE,(PHIGH)      ;X-Y AXES AT (H0,K0) IN
7C20 3E12       00070          LD      A,18            ;   THE PAGE
7C22 0E00       00080          LD      C,0
```

Listing 5 continues

Listing 5 continued

```
7C24 CDF47D                CALL   MULT          ;PAGEH=18*PHIGH
7C27 220B7E                LD     (PAGEH),HL
7C2A ED5B077E              LD     DE,(PWIDE)
7C2E 3E10                  LD     A,16
7C3Ø ØEØØ                  LD     C,Ø
7C32 CDF47D                CALL   MULT          ;PAGEW=16*PWIDE
7C35 22097E                LD     (PAGEW),HL
7C38 7D                    LD     A,L
7C39 4C                    LD     C,H
7C3A ED5BØ57E              LD     DE,(PHIGH)
7C3E CDF47D                CALL   MULT          ;3*(PAGEW)*(PHIGH) EQUALS
7C41 EB                    EX     DE,HL         ;  THE NUMBER OF BYTES IN
7C42 3EØ3                  LD     A,3           ;THE PAGE
7C44 ØEØØ                  LD     C,Ø
7C46 CDF47D                CALL   MULT          ;(HL)=3*(PAGEW)*(PHIGH)
7C49 44                    LD     B,H
7C4A 4D                    LD     C,L
7C4B 21ØØ7Ø          ZERO  LD     HL,PAGE       ;ZERO THE PAGE
7C4E 36ØØ            ZLOOP LD     (HL),Ø
7C5Ø 23                    INC    HL
7C51 ØB                    DEC    BC
7C52 78                    LD     A,B
7C53 B1                    OR     C
7C54 2ØF8                  JR     NZ,ZLOOP      ;SEE IF PAGE IS ZEROED
7C56 3EØØ                  LD     A,Ø
7C58 32117E                LD     (U),A         ;U MAKES LEFT HASH MARK
7C5B 3E2Ø                  LD     A,2ØH         ;   ON THE Y-AXIS
7C5D 32137E                LD     (V),A         ;V MAKES THE RIGHT HASH
7C6Ø 3AØD7E                LD     A,(HØ)        ;   MARK ON Y-AXIS
7C63 FEØØ                  CP     Ø             ;SEE IF Y-AXIS = V-AXIS
7C65 CA6D7C                JP     Z,ISEQU
7C68 3E2Ø                  LD     A,2ØH
7C6A 32117E                LD     (U),A
7C6D 2AØ97E          ISEQU LD     HL,(PAGEW)
7C7Ø 7D                    LD     A,L
7C71 4C                    LD     C,H
7C72 11Ø3ØØ                LD     DE,3
7C75 CDF47D                CALL   MULT          ;FIND 3*PAGEW=NO. SPACES
7C78 E5                    PUSH   HL            ;BETWEEN Y MARKS
7C79 3E1Ø                  LD     A,16
7C7B ØEØØ                  LD     C,Ø
7C7D ED5BØD7E              LD     DE,(HØ)
7C81 CDF47D                CALL   MULT          ;FIND 16*H
7C84 11ØØ7Ø                LD     DE,PAGE
7C87 19                    ADD    HL,DE         ;PAGE+16*H
7C88 E5                    PUSH   HL
7C89 2AØ57E                LD     HL,(PHIGH)    ;# OF Y HASH MARKS
7C8C 45                    LD     B,L
7C8D E1                    POP    HL
7C8E D1                    POP    DE
7C8F 22197E          MARKY LD     (TEMP),HL
7C92 DD2A197E              LD     IX,(TEMP)     ;IX POINTER TO Y-AXIS
7C96 3A117E                LD     A,(U)
7C99 DD77FF                LD     (IX+255),A    ;MARK LEFT OF Y-AXIS
7C9C 3A137E                LD     A,(V)
7C9F DD77Ø1                LD     (IX+1),A      ;MARK RIGHT OF Y-AXIS
7CA2 19                    ADD    HL,DE         ;FOR NEXT HASH MARK
7CA3 1ØEA                  DJNZ   MARKY
7CA5 2AØ97E                LD     HL,(PAGEW)    ;BEGIN TO MARK X-AXIS
7CA8 7D                    LD     A,L
7CA9 4C                    LD     C,H
7CAA ED5BØF7E              LD     DE,(KØ)
7CAE CDF47D                CALL   MULT          ;FIND K*PAGEW
7CB1 11Ø3ØØ                LD     DE,3
7CB4 7D                    LD     A,L
7CB5 4C                    LD     C,H
7CB6 CDF47D                CALL   MULT          ;(HL)=PAGE+3*K*(PAGEW)
```

Listing 5 continues

```
7CB9 110070   00760         LD    DE,PAGE      ;  POINTS X-AXIS
7CBC 19       00770         ADD   HL,DE
7CBD 2B       00780         DEC   HL
7CBE E5       00790         PUSH  HL
7CBF 110F00   00800         LD    DE,15
7CC2 A7       00810         AND   A
7CC3 ED52     00820         SBC   HL,DE
7CC5 111000   00830         LD    DE,16
7CC8 E5       00840         PUSH  HL           ;(HL)=PAGE+3*K*(PAGEW)-16
7CC9 2A077E   00850         LD    HL,(PWIDE)   ; LEFTMOST PART OF X-AXIS
7CCC 45       00860         LD    B,L
7CCD 05       00870         DEC   B            ;# HASH MARKS ON X-AXIS
7CCE E1       00880         POP   HL           ;IS (PAGEW)-1
7CCF 7E       00890 MARKX   LD    A,(HL)
7CD0 F602     00900         OR    2            ;MAKE DOT OVER X-AXIS
7CD2 77       00910         LD    (HL),A
7CD3 E5       00920         PUSH  HL
7CD4 D5       00930         PUSH  DE
7CD5 ED5B097E 00940         LD    DE,(PAGEW)
7CD9 19       00950         ADD   HL,DE
7CDA 22197E   00960         LD    (TEMP),HL
7CDD DD2A197E 00970         LD    IX,(TEMP)
7CE1 DD7E00   00980         LD    A,(IX)
7CE4 F620     00990         OR    20H          ;MAKE DOT UNDER X-AXIS
7CE6 DD7700   01000         LD    (IX),A
7CE9 D1       01010         POP   DE
7CEA E1       01020         POP   HL
7CEB A7       01030         AND   A
7CEC ED52     01040         SBC   HL,DE        ;MOVE LEFT FOR NEXT MARK
7CEE 10DF     01050         DJNZ  MARKX
7CF0 2A057E   01060 YAXIS   LD    HL,(PHIGH)
7CF3 7D       01070         LD    A,L
7CF4 4C       01080         LD    C,H
7CF5 110300   01090         LD    DE,3
7CF8 CDF47D   01100         CALL  MULT         ;3*(PHIGH)=# OF BYTES FOR
7CFB E5       01110         PUSH  HL           ;   Y-AXIS
7CFC 3E10     01120         LD    A,16
7CFE 0E00     01130         LD    C,0
7D00 ED5B0D7E 01140         LD    DE,(H0)
7D04 CDF47D   01150         CALL  MULT         ;FIND 16*H
7D07 110070   01160         LD    DE,PAGE
7D0A 19       01170         ADD   HL,DE        ;PAGE+3*(H) POINTS TO
7D0B D1       01180         POP   DE           ; START OF Y-AXIS
7D0C 43       01190         LD    B,E
7D0D ED5B097E 01200         LD    DE,(PAGEW)
7D11 363F     01210 YLOOP   LD    (HL),3FH     ;3FH MAKES COLUMN 6 DOTS
7D13 19       01220         ADD   HL,DE
7D14 10FB     01230         DJNZ  YLOOP
7D16 2A097E   01240         LD    HL,(PAGEW)   ;START OF X-AXIS
7D19 2B       01250         DEC   HL
7D1A 44       01260         LD    B,H
7D1B 4D       01270         LD    C,L
7D1C E1       01280         POP   HL
7D1D 7E       01290 XLOOP   LD    A,(HL)
7D1E F601     01300         OR    1            ;MAKES ONE DOT FOR AXIS
7D20 77       01310         LD    (HL),A
7D21 2B       01320         DEC   HL           ;MOVE LEFT FOR NEXT DOT
7D22 0B       01330         DEC   BC           ;DECREASE COUNTER
7D23 78       01340         LD    A,B
7D24 B1       01350         OR    C
7D25 FE00     01360         CP    0
7D27 C21D7D   01370         JP    NZ,XLOOP     ;SEE IF DONE WITH X-AXIS
7D2A C9       01380         RET                ;END OF AXES
7D2B 00       01390         NOP                ;BEGIN OF PLOT
7D2C 2A137E   01400 PLOT    LD    HL,(V)       ;LOAD V INTO HL
7D2F 7C       01410         LD    A,H          ;TO PLOT (U,V)--DIVIDE V
7D30 4D       01420         LD    C,L          ;BY 6, GET Q AND R THEN
```

Listing 5 continues

ICS/BAS begins at line 85. Note, as mentioned above, that H0 and K0 refer to the placement of the axes. Line 85 gives the prompt

```
CHOICES (1) INPUT CONIC (2) DRAW
  (3) END
```

Entering a 3 stops the program and returns to Basic's ready prompt. A reply of 2 sends the page to the printer and returns to line 85. The first option lets you select the type of conic and enter it into the page. This case also returns to the prompt at line 85.

Line 90 is a three-way branch; the action taken depends upon the response to line 85. Lines 100–125 determine the type of conic to be graphed. The options are line, parabola, circle, ellipse, or hyperbola. Line 130 is an error trap to ensure a proper reply at this point of the program.

Line 135 is a five-way branch, and the action taken depends upon the response made to line 125. Since each branch is essentially a subprogram to graph one of the conics, I will discuss each branch individually. For each option, the final prompt is for the domain. When necessary, the program restricts the domain to values of x for which the function is defined. The final step of each option is a jump to line 85. Lines 145–305 branch for a straight line.

Selecting the line option initiates a series of prompts; the first is

```
OPTIONS FOR FORM OF LINE:
(1) TWO-POINT
(2) POINT-SLOPE
```

If the two-point form is selected, then the coordinates of the points must be entered. There is a prompt for each point and the data is entered in the form X,Y (first coordinate, comma, second coordinate, enter). If the x coordinates of both points are the same, then the line is parallel to the y axis. This is the only way such a line can be entered. If this is the case, the prompt "Line is parallel to the y axis, input range of y's" is given. In response, enter the smallest value of y, then the largest value of y (separated by a comma) for which this line is to be graphed.

If the second option is selected, there are two prompts: one for the coordinates of the point and the other for the slope of the line.

The program does not accept lines given in the y-intercept form ($y=mx+b$). This form is a special case of the point-slope form, using the slope m and the point (0,b). It also does not accept lines in the form $ax+by=c$. This form also can be entered using the point-slope form where the point is (0, c/b) and the slope is given by $m=(-a/b)$. The remainder of this branch is devoted to the task of inserting the line into the page.

Lines 315–435 branch for a parabola. Selecting the option for the parabola gives the prompt "(1) Opens Up/Down (2) Opens Left/Right." There are two forms for the equation of a parabola with its vertex at the point (H1,K1), they are $(X-H1)^2=M(Y-K1)$, which opens up if M is greater than zero or down if M is less than zero, and $(Y-K1)^2=M(X-H1)$, which opens to the right if M is greater than zero or left if M is less than zero. Next, a prompt for the values H1, K1, and M is given. Enter the three values on the same line, separated by commas. If the parabola opens left or right, it requires two functions to graph it—one for the upper half and one for the lower half. Lines 360–375 do the computations for up/down parabolas and lines 385–435 do the left/right ones.

Lines 445–550 branch for a circle or ellipse. The standard form for the circle with its center at (H1,K1) and radius R is given by $(X-H1)^2+(Y-K1)^2=R^2$.

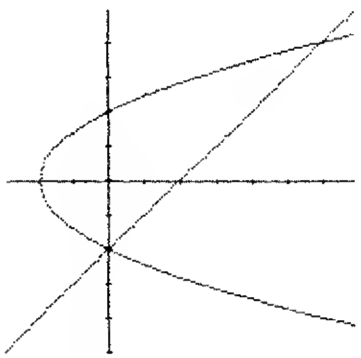
The standard form for the ellipse with its center at (H1,K1) is:

$$\frac{(X-H1)^2}{A^2}+\frac{(Y-K1)^2}{B^2}=1$$

In the equation of an ellipse, if A is greater than B, the major axis is parallel to the x axis. If A is less than B, the major axis is parallel to the y axis. If A equals B, the figure is a circle. Using these relations between A and B, I combined the computations for a circle and an ellipse into one routine. The test for which type conic is done in line 465. The first prompt in this routine is for the center: "Enter the coordinates of the center."

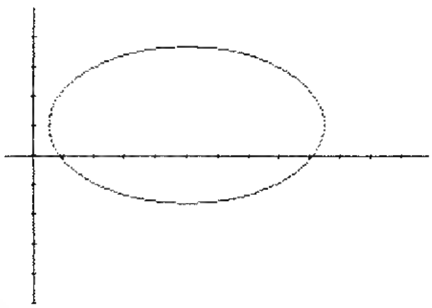
Enter the values for H1 and K1, separated by a comma. If the conic is a circle, the next data entered must be the square of the radius. This allows the use of a radius such as $\sqrt{3}$ without having to enter its decimal approximation. If the conic is an ellipse, then the squares of A and B must be entered—for the same reason the square of the radius is given. Circles and ellipses require two functions—one for the upper half and one for the lower half. The lower portion is drawn in lines 500–515 and the upper part in lines 525–545.

Lines 555–670 branch for a hyper-



PARABOLA $y^2=2(x+2)$
LINE $y=x-2$

Figure 15



```
PHIGH = PAGE HEIGHT
PWIDE = PAGE WIDTH
INPUT PHIGH? 10
INPUT PWIDE? 14
INPUT H0? 1
INPUT K0? 5
CHOICES (1) INPUT CONIC
  (2) DRAW (3) END? 1
CHOICES OF CONICS TO GRAPH:
  (1) LINE
  (2) PARABOLA
  (3) CIRCLE
  (4) ELLIPSE
  (5) HYPERBOLA
ENTER THE APPROPRIATE NUMBER? 4
ENTER THE COORDINATES OF THE
  CENTER? 5,1
ENTER A AND B SQUARED? 20,7
ENTER THE DOMAIN? -1,13
```

Figure 16

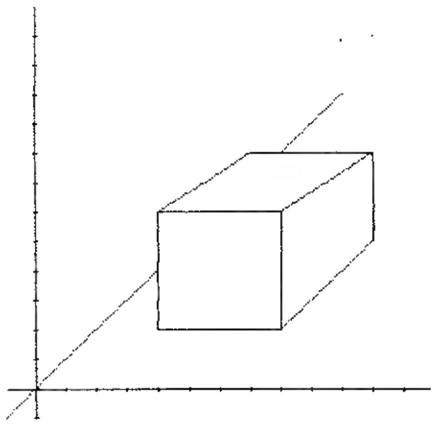


Figure 17

```
7D31 1EØ6     Ø143Ø        LD     E,6          ;SET BIT 5-R IN BYTE
7D33 16ØØ     Ø144Ø        LD     D,Ø          ;PAGE+Q*(PAGEW)+U
7D35 CDDE7D   Ø145Ø        CALL   DIVIDE
7D38 57       Ø146Ø        LD     D,A
7D39 59       Ø147Ø        LD     E,C
7D3A ED53157E Ø148Ø        LD     (QUOT),DE    ;SAVE Q
7D3E 22177E   Ø149Ø        LD     (RMDER),HL   ;SAVE R
7D41 3AØ97E   Ø15ØØ        LD     A,(PAGEW)
7D44 ØEØØ     Ø151Ø        LD     C,Ø          ;  DIVISOR ON DE
7D46 ED5B157E Ø152Ø        LD     DE,(QUOT)    ;ON EXIT QUOTIENT IN AD
7D4A CDF47D   Ø153Ø        CALL   MULT         ;  REMAINDER IN HL
7D4D ED4B117E Ø154Ø        LD     BC,(U)
7D51 Ø9       Ø155Ø        ADD    HL,BC
7D52 Ø1ØØ7Ø   Ø156Ø        LD     BC,PAGE
7D55 Ø9       Ø157Ø        ADD    HL,BC        ;ADD ADDR OF PAGE
7D56 22197E   Ø158Ø        LD     (TEMP),HL
7D59 DD2A197E Ø159Ø        LD     IX,(TEMP)
7D5D 21Ø5ØØ   Ø16ØØ        LD     HL,5
7D6Ø ED5B177E Ø161Ø        LD     DE,(RMDER)   ;FIND 5 MINUS RMDER
7D64 A7       Ø162Ø        AND    A            ;DETERMINE WHICH BIT
7D65 ED52     Ø163Ø        SBC    HL,DE        ;   TO SET
7D67 16Ø1     Ø164Ø        LD     D,1
7D69 7D       Ø165Ø        LD     A,L
7D6A B7       Ø166Ø        OR     A
7D6B 28Ø5     Ø167Ø        JR     Z,RDONE      ;GO NO MORE ROTATES
7D6D 47       Ø168Ø        LD     B,A
7D6E CB22     Ø169Ø RLOOP  SLA    D            ;ROTATE TO SET BIT
7D7Ø 1ØFC     Ø17ØØ        DJNZ   RLOOP
7D72 DD7EØØ   Ø171Ø RDONE  LD     A,(IX)       ;GET CONTENTS OF BYTE
7D75 B2       Ø172Ø        OR     D            ;INSERT NEW INFO IN BYTE
7D76 DD77ØØ   Ø173Ø        LD     (IX),A       ;STORE BACK INTO BYTE
7D79 C9       Ø174Ø        RET                 ;END OF PLOT
7D7A ØØ       Ø175Ø        NOP                 ;BEGIN OF PRINT
7D7B 3EØ3     Ø176Ø PRINT  LD     A,3          ;**
7D7D CDD37D   Ø177Ø        CALL   OUTCHR       ;PUT 445 INTO GRAPHICS
7D8Ø ØEØØ     Ø178Ø        LD     C,Ø
7D82 ED5BØ57E Ø179Ø        LD     DE,(PHIGH)
7D86 CDF47D   Ø18ØØ        CALL   MULT
7D89 2B       Ø181Ø PLOOP  DEC    HL           ;COUNTER VERTICALLY IS
7D8A 22117E   Ø182Ø        LD     (U),HL       ;(U)=3*(PHIGH)-1
7D8D EB       Ø183Ø        EX     DE,HL
7D8E 2AØ97E   Ø184Ø        LD     HL,(PAGEW)
7D91 7D       Ø185Ø        LD     A,L
7D92 4C       Ø186Ø        LD     C,H
7D93 CDF47D   Ø187Ø        CALL   MULT         ;(HL)=U*(PAGEW)
7D96 11ØØ7Ø   Ø188Ø        LD     DE,PAGE
7D99 19       Ø189Ø        ADD    HL,DE        ;(HL)=PAGE+U*(PAGEW)
7D9A 11ØØØØ   Ø19ØØ        LD     DE,Ø         ;IS THE COUNTER ACROSS
7D9D 7E       Ø191Ø GETCHR LD     A,(HL)
7D9E FEØ3     Ø192Ø        CP     3            ;**IF CHAR=3,PRINT TWICE
7DAØ CCD37D   Ø193Ø        CALL   Z,OUTCHR     ;SEND CHARACTER
7DA3 CDD37D   Ø194Ø        CALL   OUTCHR       ;SEND CHARACTER
7DA6 13       Ø195Ø        INC    DE           ;INCREASE POINTER
7DA7 23       Ø196Ø        INC    HL           ;INCREASE POINTER
7DA8 E5       Ø197Ø        PUSH   HL           ;SAVE POINTER
7DA9 2AØ97E   Ø198Ø        LD     HL,(PAGEW)   ;CHECK-SEE IF LINE DONE
7DAC 2B       Ø199Ø        DEC    HL
7DAD A7       Ø2ØØØ        AND    A
7DAE ED52     Ø2Ø1Ø        SBC    HL,DE
7DBØ 7C       Ø2Ø2Ø        LD     A,H
7DB1 B5       Ø2Ø3Ø        OR     L
7DB2 E1       Ø2Ø4Ø        POP    HL
7DB3 C29D7D   Ø2Ø5Ø        JP     NZ,GETCHR    ;CONTINUE IF NOT DONE
7DB6 3EØ3     Ø2Ø6Ø        LD     A,3          ;**SEND END OF LINE SGNAL
7DB8 CDD37D   Ø2Ø7Ø        CALL   OUTCHR
7DBB 3EØB     Ø2Ø8Ø        LD     A,ØBH        ;**
7DBD CDD37D   Ø2Ø9Ø        CALL   OUTCHR
```

Listing 5 continues

Listing 5 continued

```
7DC0 2A117E   02100        LD     HL,(U)      ;CHECK-SEE IF PAGE DONE
7DC3 7C       02110        LD     A,H
7DC4 B5       02120        OR     L
7DC5 C2897D   02130        JP     NZ,PLOOP    ;CONTINUE IF NOT DONE
7DC8 3E03     02140 DONE   LD     A,3         ;**SIGNAL TO EXIT GRAPHICS
7DCA CDD37D   02150        CALL   OUTCHR
7DCD 3E02     02160        LD     A,2         ;**
7DCF CDD37D   02170        CALL   OUTCHR
7DD2 C9       02180        RET                ;RET TO CALLING PROG
7DD3 F5       02190 OUTCHR PUSH   AF          ;SUBR TO SEND CHAR
7DD4 CDD105   02200 CHCKP  CALL   5D1H        ;CHECK STATUS OF PRINTER
7DD7 20FB     02210        JR     NZ,CHCKP    ;WAIT IF NOT READY
7DD9 F1       02220        POP    AF
7DDA 32E837   02230        LD     (37E8H),A   ;SEND IT
7DDD C9       02240        RET                ;RET TO PRINT
7DDE 210000   02250 DIVIDE LD     HL,0        ;FROM ZAKS
7DE1 0610     02260        LD     B,16        ;ON ENTRY DIVIDEND IN AC
7DE3 CB11     02270 DLOOP  RL     C           ;         DIVISOR IN HL
7DE5 17       02280        RLA                ;ON EXIT QUOTIENT IN AC
7DE6 ED6A     02290        ADC    HL,HL       ;        REMAINDER IN HL
7DE8 ED52     02300        SBC    HL,DE
7DEA 3001     02310        JR     NC,$+3
7DEC 19       02320        ADD    HL,DE
7DED 3F       02330        CCF
7DEE 10F3     02340        DJNZ   DLOOP
7DF0 CB11     02350        RL     C
7DF2 17       02360        RLA
7DF3 C9       02370        RET                ;END OF DIVIDE SUBROUTINE
7DF4 0610     02380 MULT   LD     B,16        ;FROM ZAKS
7DF6 210000   02390        LD     HL,0        ;ON ENTRY (A)=LSB MULTCND
7DF9 CB39     02400 MLOOP  SRL    C           ;         (C)=MSB MULTCND
7DFB 1F       02410        RRA                ;         (DE)=MULTIPLIER
7DFC 3001     02420        JR     NC,MLOOP1   ;ON EXIT (HL)=PRODUCT
7DFE 19       02430        ADD    HL,DE
7DFF EB       02440 MLOOP1 EX     DE,HL
7E00 29       02450        ADD    HL,HL
7E01 EB       02460        EX     DE,HL
7E02 10F5     02470        DJNZ   MLOOP
7E04 C9       02480        RET          ;END OF MULT SUBROUTINE
7E05          02490 PHIGH  EQU    $
0002          02500        DEFS   2
7E07          02510 PWIDE  EQU    $
0002          02520        DEFS   2
7E09          02530 PAGEW  EQU    $
0002          02540        DEFS   2
7E0B          02550 PAGEH  EQU    $
0002          02560        DEFS   2
7E0D          02570 H0     EQU    $
0002          02580        DEFS   2
7E0F          02590 K0     EQU    $
0002          02600        DEFS   2
7E11          02610 U      EQU    $
0002          02620        DEFS   2
7E13          02630 V      EQU    $
0002          02640        DEFS   2
7E15          02650 QUOT   EQU    $
0002          02660        DEFS   2
7E17          02670 RMDER  EQU    $
0002          02680        DEFS   2
7E19          02690 TEMP   EQU    $
0002          02700        DEFS   2
7000          02710 PAGE   EQU    7000H
0000          02720        END
00000 TOTAL ERRORS
```

bola. The hyperbola has two possible forms for its equation:

TWO OPTIONS:
(1) REAL AXIS PARALLEL TO X AXIS
(2) REAL AXIS PARALLEL TO Y AXIS

The equation for the real axis parallel to the x axis is

$$\frac{(X-H1)^2}{A^2}-\frac{(Y-K1)^2}{B^2}=1$$

and for the real axis parallel to the y axis is

$$-\frac{(X-H1)^2}{A^2}+\frac{(Y-K1)^2}{B^2}=1$$

After selecting option 1 or 2, enter the center's coordinates. In the case of the hyperbola, the center is the midpoint of the line joining the two vertices of the hyperbola. Then, as was the case for the ellipse, enter the squares of the constants A and B. It takes two functions to draw a hyperbola that opens up and down, namely one for the upper portion and one for the lower portion. For a hyperbola that opens left and right, it takes four functions—upper and lower parts of both branches. Obviously, this is the slowest type of function to graph. Using variables M1 through M4, as defined in lines 595 and 600, I condensed the computational portions of this segment of the program. These same constants also were used to redefine the domain of the hyperbola that opens left and right, if necessary.

Consider the following example. Given the equation for the ellipse: $X^2 + 4Y^2 - 2X + 16Y = 7$. First, put the equation into standard form by collecting the terms as shown, then completing the square:

$$(x^2 - 2x \quad\;) + 4(y^2 + 4y \quad\;) = 7$$
$$(x^2 - 2x + 1) + 4(y^2 + 4y + 4) = 7 + 1 + 16$$
$$(x - 1)^2 + 4(y + 2)^2 = 24$$

Dividing by 24 gives the standard form

$$\frac{(x-1)^2}{24} + \frac{(y+2)^2}{6} = 1.$$

This equation, in standard form, gives the values of $(H1,K1) = 1, -2)$, $A^2 = 24$, and $B^2 = 6$. Use the values $H0 = 7$ and $K0 = 6$ to place the x-y axes. Figure 12 shows the prompts, the answers, and the graph of this ellipse. The time the program takes to insert the ellipse into the page is about 80 seconds. Figures 13–17 give various examples of graphing conic sections using CONICS/BAS.

The changes necessary for a 16K Level II machine are not difficult. These changes result from the difference between the USR calls in Disk Basic and Level II Basic, and the lack of the DEFFN command. The program segment given in Listing 5 modifies Tigergraph to Tiger16K. (Insert or change the lines as necessary.) The portion that replaces Getu is a subprogram designed to call the appropriate routines. In particular, when the command USR(N) is given from Basic, the argument N (N must be 0, 1, or 2) is entered into Tiger16K with the command CALL 0A7FH. If N is zero, the Axes subroutine is called. If N is one, the Plot subroutine is called. Otherwise, N must be two and the Print subroutine is called.

If any other changes are made, the addresses that are POKEd into the Basic program must be changed to match the resulting addresses of the variables H, K, PHIGH, and PWIDE when Tiger16K is compiled.

Program Listing 6 is equivalent to Program Listing 1, except it is compatible with Level II Basic. The differences are: Line 10 POKEs the starting address of the Tiger16K. Line 20 is deleted. Line 90 is changed to Z = USR(0), where the argument being zero causes the page to be zeroed and the coordinate axes to be drawn.

Line 100 is deleted. Line 130 Z = USR3(U) is replaced by two POKE statements. These statements POKE U into the appropriate memory locations. Line 140 defines the function to be graphed and changes it to the u-v coordinate system. Lines 150 and 200 replace Z = USR1(V) by two POKE statements. These statements POKE V into the appropriate memory page locations.

With page's starting address at 7000H, the page size must be less than or equal to 0C00H, or 3,072 in decimal. The page size (number of bytes) is computed by 3(PHIGH)(16)(PWIDE). Since this number should not exceed 3,072, (PHIGH) × (PWIDE) must be less than or equal to 64.

Tigergraph can be adapted to a different printer, as long as it operates similarly to the Paper Tiger. (Compare the control codes and Figs. 1 and 2 with your owner's manual.) All output is done in the Print routine; I have marked the lines that are Tiger control codes with a double asterisk in the comment column. One word of caution: If changes are made in Tigergraph, then it might be necessary to change the POKE statements that correspond to H, K, PHIGH, and PWIDE, to the new addresses for those variables.

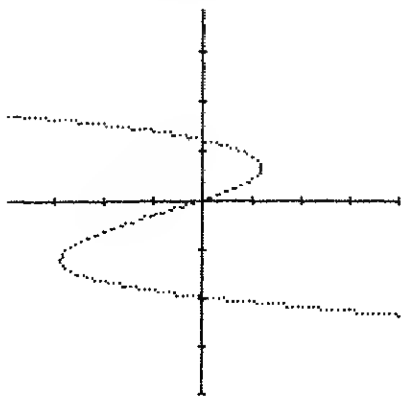


Figure 19

Program Listing 6

```
10 POKE 16526,01:POKE 16527,124
30 DEFINT I,J,T,U,V
40 I6=16:I8=18:P5=.5
50 INPUT"INPUT H";H:POKE 32269,H:POKE 32270,0
60 INPUT"INPUT K";K:POKE 32271,K:POKE 32272,0
70 INPUT"INPUT PHIGH";I:POKE32261,I:POKE32262,0:V5=I8*I
80 INPUT"INPUT PWIDE";I:POKE32263,I:POKE32264,0
90 Z=USR(0)
110 INPUT"ENTER THE DOMAIN";A1,A2
120 I1=INT(I6*(A1+H)+P5):I2=INT(I6*(A2+H)+P5)
130 FOR U=I1 TO I2:POKE 32273,U:POKE 32274,0
140 V=INT(I8*(K+2*(U/I6-H)-3)+P5)
150 IFV<0 OR V> V5 THEN 210ELSEPOKE32275,V:POKE32276,0:Z=USR(1)
160 IF I=I1 THEN 210
170 IF ABS(T-V)<3 THEN 210
180 IF T>V THEN J1=V+1:J2=T-1
190 IF T<V THEN J1=T+1:J2=V-1
200 FOR I3=J1 TO J2 STEP 2:POKE32275,V:POKE32276,0:Z=USR(1)
210 T=V
220 NEXT U
225 Z=USR(2)
230 END
```

Part II

Tiger Graphics with Pascal-80

Pascal-80 is a standard version of Pascal. It has omitted a few features of standard Pascal, but the omissions are offset by several extra, very useful features.

The system operates under TRSDOS and NEWDOS40, but not NEWDOS80. Pascal-80 is distributed through TSE-Hardside, 6 South Street, Milford, NH 03055 and costs $99.95.

The manual provided with Pascal-80 is brief; in fact, it is too brief. Because of the minimal documentation, the best way to become familiar with Pascal-80 is to select a book on standard Pascal and to try each feature. However, even this method has its problems. For example, in the section describing the extensions to standard Pascal, the Call function is described:

CALL(ADDRESS,VALUE) places a value (0–255) into the A register, and calls the address. It returns the contents of the A register after the call (type INTEGER).

However, the manual fails to give the very important information that if a machine-language subprogram uses the

```
0000 PROGRAM TABS;
0003 VAR X: ARRAY(.1..55.) OF CHAR;
0016 BEGIN
001B  CLS;
001C  X:='1234567890123456789012345678901234567890123456789012345';
005A   WRITELN(X);
006B   WRITELN(CHR(201),'A',CHR(201),'B',CHR(201),'C')
0089 END.
```

Program Listing 7

Program Listing 8

```
                ;TIGERPAS
                ;BARBARA CLINGER
                ;DEPARTMENT OF MATHEMATICS
                ;WHEATON COLLEGE
                ;NORTON, MASS. 02766
                ;
                ;TO OPERATE FROM PASCAL-80 PROGRAMS
                ;JULY 2, 1982
                ;
F500                            ORG     0F500H
F500 C321F5                     JP      START
F503                    PHIGH   EQU     $           ;NO. OF UNITS FOR Y-AXIS
F505                    PWIDE   EQU     $+2         ;NO. OF UNITS FOR X-AXIS
F507                    H0      EQU     $+4         ;FIRST COORDINATE OF ORIGIN
F509                    K0      EQU     $+6         ;SECOND COORDINATE OF ORIGIN
F50B                    U       EQU     $+8         ;1ST COORD. OF POINT TO PLOT
F50D                    V       EQU     $+10        ;2ND COORD. OF POINT TO PLOT
F50F                    PAGEH   EQU     $+12        ;#OF DOTS FOR Y-AXIS
F511                    PAGEW   EQU     $+14        ;# OF DOTS FOR X-AXIS
F513                    TEMP    EQU     $+16
F515                    QUOT    EQU     $+18
F517                    RMDER   EQU     $+20
F519                    LINES   EQU     $+22        ;# OF LINES IN A PAGE
0012                    I8      EQU     18          ;# OF DOTS PER (1/4) IN. VERTICAL
0010                    I6      EQU     16          ;# OF DOTS PER (1/4) IN HORIZONTAL
0006                    DOTS    EQU     6           ;# DOTS PER 1 PASS PRINT HEAD
D000                    PAGE    EQU     0D000H
001E                            DEFS    30
                ; FIRST PART OF PROGRAM DECIDES WHICH PORTION OF PROGRAM
                ; TO EXECUTE--  0=ZERO THE PAGE
                ;               1=DRAW THE AXES
                ;               2=PLOT POINT (U,V)
                ;               3=SEND PAGE TO PRINTER
F521 FE00               START   CP      0
F523 C22AF5                     JP      NZ,NOTZER       ;JUMP IF NOT ZERO
F526 CD40F5                     CALL    ZERO
F529 C9                         RET
```

Listing 8 continues

```
F52A FE01       NOTZER  CP      1               ;COMPARE IT TO ONE
F52C C233F5             JP      NZ,NOTONE       ;JUMP IF NOT ONE
F52F CD90F5             CALL    AXES
F532 C9                 RET
F533 FE02       NOTONE  CP      2               ;COMPARE IT TO TWO
F535 C23CF5             JP      NZ,NOTTWO
F538 CD48F6             CALL    PLOT
F53B C9                 RET
F53C CD9EF6     NOTTWO  CALL    PRINT           ;MUST BE THREE
F53F C9                 RET
F540 00         ZERO    NOP                     ;START OF ZERO
                ;ZERO SETS ALL MEMORY IN PAGE TO ZERO. IT ALSO CALCULATES
                ;SOME OF THE VARIABLES FOR THE REST OF THE PROGRAM. ZERO
                ;MUST BE CALLED BEFORE USING OTHER ROUTINES IN THIS
                ;PROGRAM.
F541 ED5B03F5           LD      DE,(PHIGH)
F545 3E12               LD      A,I8
F547 0E00               LD      C,0
F549 CD17F7             CALL    MULT            ;FIND PAGEH=I8*PHIGH
F54C 220FF5             LD      (PAGEH),HL
F54F ED5B05F5           LD      DE,(PWIDE)
F553 3E10               LD      A,I6
F555 0E00               LD      C,0
F557 CD17F7             CALL    MULT            ;FIND PAGEW=I6*PWIDE
F55A 2211F5             LD      (PAGEW),HL
F55D 2A0FF5             LD      HL,(PAGEH)      ;FIND NUMBER OF LINES
F560 7C                 LD      A,H             ;IN PAGE=# DOTS VERTICAL
F561 4D                 LD      C,L             ;DIVIDED BY DOTS, ROUND
F562 110600             LD      DE,DOTS         ;UP IF NECESSARY
F565 CD01F7             CALL    DIVIDE
F568 57                 LD      D,A
F569 59                 LD      E,C             ;QUOT INTO DE
F56A 7D                 LD      A,L             ;SEE IF RMDER IS ZERO
F56B B4                 OR      H
F56C FE00               CP      0
F56E CA72F5             JP      Z,LINER         ;IF RMDER=0, LINES=QUOT
F571 13                 INC     DE              ;OTHERWISE LINES=QUOT+1
F572 ED5319F5   LINER   LD      (LINES),DE
F576 2A11F5             LD      HL,(PAGEW)      ;FIND NO. OF BYTES IN
F579 7D                 LD      A,L             ;PAGE = (LINES)*(PAGEW)
F57A 4C                 LD      C,H
F57B ED5B19F5           LD      DE,(LINES)
F57F CD17F7             CALL    MULT            ;(HL)=(LINES)*(PAGEW)
F582 44                 LD      B,H
F583 4D                 LD      C,L
F584 2100D0             LD      HL,PAGE         ;ZERO THE PAGE
F587 3600       ZLOOP   LD      (HL),0
F589 23                 INC     HL
F58A 0B                 DEC     BC
F58B 78                 LD      A,B
F58C B1                 OR      C
F58D 20F8               JR      NZ,ZLOOP        ;SEE IF PAGE IS ZEROED
F58F C9                 RET
F590 00         AXES    NOP                     ;BEGIN OF AXES
                ;THIS DRAWS THE COODINATE AXES WITH ORIGIN (H0,K0)
                ;THE X-AXIS IS I8*K0 UNITS UP FROM BOTTOM OF PAGE
                ;THE Y-AXIS IS I6*H0 UNITS FROM LEFT OF PAGE
                ;HASH MARKS ARE MADE EVERY I6 DOTS ON X-AXIS AND EVERY
                ;I8 DOTS ON THE Y-AXIS
F591 ED5B09F5   XAXIS   LD      DE,(K0)
F595 3E12               LD      A,I8
F597 0E00               LD      C,0
F599 CD17F7             CALL    MULT
F59C 220DF5             LD      (V),HL          ;(HL)=I8*K0=UNITS UP
F59F 2A11F5             LD      HL,(PAGEW)
F5A2 2B                 DEC     HL
```

Listing 8 continues

IX register, it is necessary to save the value in the register upon entry into the subprogram and to restore it before returning to the calling program.

Another item not in the manual is the ability to use CHR to perform a tab—similar to its use in Tiny Pascal and Basic. Specifically, a tab on the video screen is accomplished by using the control codes 192–255 as follows: If x is an integer from 192–255 (inclusive), then the command CHR(x) in a write statement causes the cursor to be moved X spaces to the right. That is, CHR(192) does not move the cursor, CHR(193) moves the cursor one place right, CHR(194) moves it two places to the right, and so forth. In Tabs (Program Listing 7) an example of using tabs is given. It illustrates printing in columns 10, 20, and 30.

At the left of each line in the Pascal-80 listing there is a hexadecimal number. It gives the number of bytes of code compiled to that point. Although these are not real line numbers, at times I will refer to specific lines in a program using these numbers.

Changes to Tigergraph

The changes to adapt Tigergraph to be used with a Pascal-80 program result from the differences in the Call commands between Basic and Pascal-80. Recall that in the Level II and Disk Basic, the argument of the Call command must be in the range from −32,786 to 32,767 inclusive, and it is passed into the HL register pair by the special ROM Call—Call 0A7FH. In Pascal-80 the argument must be in the range from 0–255 inclusive, and it is automatically placed in the A register.

In the Tigerpas listing (Program Listing 8), the variables are defined in the first few lines of the program. The short routine immediately following the definitions of the variables gives Tigerpas a single entry point. The origin of Tigerpas is 0F500H (decimal 62,720) and this address is used in the Call command from a Pascal-80 program. Just as in a POKE statement in Basic, when an address is greater than 32767, the address is calculated by −(65536−real address). Thus, the address used in a Pascal-80 program for Tigerpas is −(65536−62720)= −2816.

In the Pascal-80 programs, the constant TIGER= −2816 is defined and the Call to Tigerpas has the form Z:= CALL(TIGER,N) 0⩽N⩽3. The argument N is placed in the A register and Tigerpas begins execution at Start. The value of N determines which subroutine in Tigerpas is to be used.

If N=0, the Zero subprogram is called and the memory-page block is set to zero. (Page is defined in Part 1 to be the memory area for the graph.) This routine also initializes variables for the remaining subroutines and must be called first.

If N=1, the Axes subprogram is executed and the x-y coordinate axes are inserted into page. Zero and Axes were one subroutine in Tigergraphics.

If N=2, the Plot subprogram is executed and a point is inserted into the page. Prior to making this Call, the values of U and V must be POKEd into their respective locations in memory. The ROM Call 022CH has been added to Plot. This ROM routine blinks an asterisk on the screen in the upper right hand corner of the monitor. The purpose of this call is to provide a signal during the execution that the program is running and has not wandered off into the nether world. Care should be used with this ROM Call since it uses all registers.

If N=3, the Print subprogram is executed and the page is sent to the printer.

The last chances in Tigergraph were made to adapt Tigerpas so it could be used with other dot-matrix printers with graphics capabilities. These changes allow flexibility in the number of dots per unit on the x-y coordinate axes and the number of vertical dots printed by a single pass of the print head.

Memory Allocations

Pascal-80 resides in memory from 21,760–40,639 (5500H–9EBFH). Programs are stored beginning at 40,704 (9F00H) in what I assume is the text-editor buffer area, and compiled programs are stored immediately after that area. The placement of the Page and the machine-language program Tigerpas is made in such a way to maximize the memory available to Pascal-80 programs. In particular, Tigerpas begins at 62,720 (F500H) and the page starts at 53,248 (D000H). This choice for the page results from the memory requirement for a 14-by-14 page (9,408 bytes).

If these allocations result in problems for the Pascal programs, there are two options:

Reduce the size of the page and in Tigerpas equate page to the appropriate larger number, or

Save and run the Pascal programs using the W and X commands. The W command from the monitor saves a machine-code "object" program file, and the X command executes an object pro-

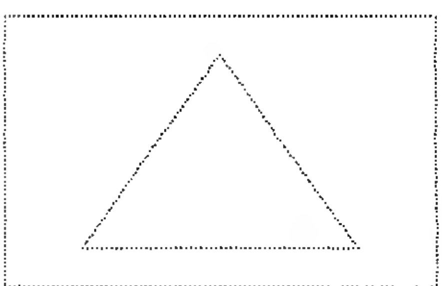


Figure 18

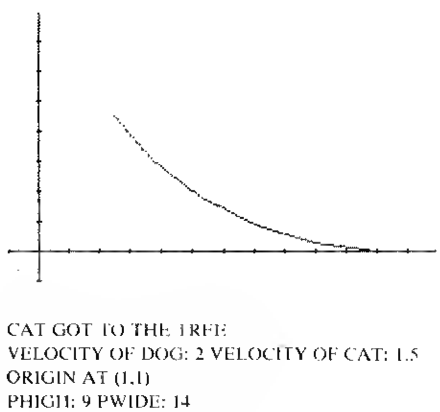


Figure 20

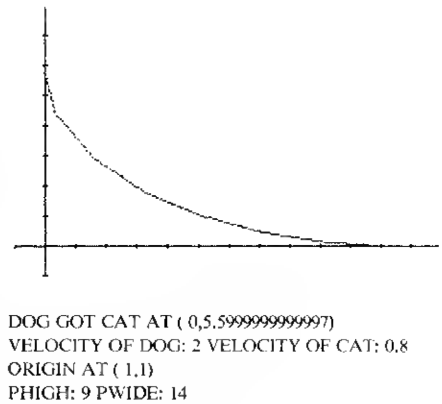


Figure 21

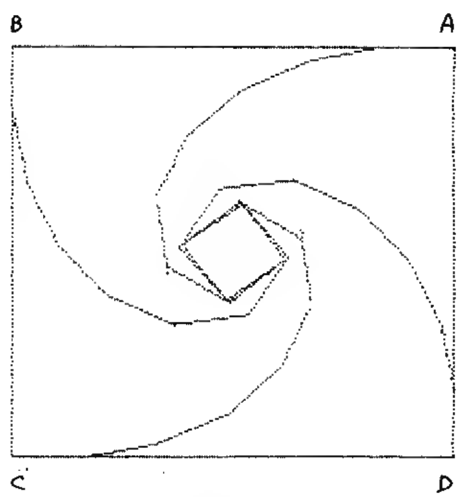


Figure 22

```
F5A3 220BF5    Ø1Ø4Ø          LD      (U),HL          ;U RANGES FROM (PAGEW) TO
F5A6 211ØØØ    Ø1Ø5Ø          LD      HL,I6           ;ZERO FOR X-AXIS
F5A9 2B        Ø1Ø6Ø          DEC     HL
F5AA E5        Ø1Ø7Ø          PUSH    HL
F5AB CD48F6    Ø1Ø8Ø XLOOP    CALL    PLOT
F5AE E1        Ø1Ø9Ø          POP     HL
F5AF 7D        Ø11ØØ          LD      A,L
F5BØ FEØØ      Ø111Ø          CP      Ø
F5B2 CAB9F5    Ø112Ø          JP      Z,LOOPX
F5B5 2B        Ø113Ø          DEC     HL
F5B6 C3D9F5    Ø114Ø          JP      TESTU
F5B9 2AØDF5    Ø115Ø LOOPX    LD      HL,(V)
F5BC 2B        Ø116Ø          DEC     HL
F5BD 22ØDF5    Ø117Ø          LD      (V),HL
F5CØ CD48F6    Ø118Ø          CALL    PLOT
F5C3 2AØDF5    Ø119Ø          LD      HL,(V)
F5C6 23        Ø12ØØ          INC     HL
F5C7 23        Ø121Ø          INC     HL
F5C8 22ØDF5    Ø122Ø          LD      (V),HL
F5CB CD48F6    Ø123Ø          CALL    PLOT
F5CE 2AØDF5    Ø124Ø          LD      HL,(V)
F5D1 2B        Ø125Ø          DEC     HL
F5D2 22ØDF5    Ø126Ø          LD      (V),HL
F5D5 211ØØØ    Ø127Ø          LD      HL,I6
F5D8 2B        Ø128Ø          DEC     HL
F5D9 E5        Ø129Ø TESTU    PUSH    HL
F5DA 2AØBF5    Ø13ØØ          LD      HL,(U)
F5DD 7D        Ø131Ø          LD      A,L
F5DE B4        Ø132Ø          OR      H
F5DF FEØØ      Ø133Ø          CP      Ø
F5E1 CAEBF5    Ø134Ø          JP      Z,ENDX
F5E4 2B        Ø135Ø          DEC     HL
F5E5 22ØBF5    Ø136Ø          LD      (U),HL
F5E8 C3ABF5    Ø137Ø          JP      XLOOP
F5EB E1        Ø138Ø ENDX     POP     HL
F5EC ED5BØ7F5  Ø139Ø YAXIS    LD      DE,(HØ)
F5FØ 3E1Ø      Ø14ØØ          LD      A,I6
F5F2 ØEØØ      Ø141Ø          LD      C,Ø
F5F4 CD17F7    Ø142Ø          CALL    MULT
F5F7 22ØBF5    Ø143Ø          LD      (U),HL          ;(HL)=I6*HØ=UNITS LEFT
F5FA 2AØFF5    Ø144Ø          LD      HL,(PAGEH)      ;V RANGES FROM (PAGEH) TO
F5FD 2B        Ø145Ø          DEC     HL              ;ZERO FROM Y-AXIS
F5FE 22ØDF5    Ø146Ø          LD      (V),HL
F6Ø1 2112ØØ    Ø147Ø          LD      HL,I8
F6Ø4 2B        Ø148Ø          DEC     HL
F6Ø5 E5        Ø149Ø          PUSH    HL
F6Ø6 CD48F6    Ø15ØØ YLOOP    CALL    PLOT
F6Ø9 E1        Ø151Ø          POP     HL
F6ØA 7D        Ø152Ø          LD      A,L
F6ØB FEØØ      Ø153Ø          CP      Ø
F6ØD CA14F6    Ø154Ø          JP      Z,LOOPY
F61Ø 2B        Ø155Ø          DEC     HL
F611 C334F6    Ø156Ø          JP      TESTV
F614 2AØBF5    Ø157Ø LOOPY    LD      HL,(U)
F617 2B        Ø158Ø          DEC     HL
F618 22ØBF5    Ø159Ø          LD      (U),HL
F61B CD48F6    Ø16ØØ          CALL    PLOT
F61E 2AØBF5    Ø161Ø          LD      HL,(U)
F621 23        Ø162Ø          INC     HL
F622 23        Ø163Ø          INC     HL
F623 22ØBF5    Ø164Ø          LD      (U),HL
F626 CD48F6    Ø165Ø          CALL    PLOT
F629 2AØBF5    Ø166Ø          LD      HL,(U)
F62C 2B        Ø167Ø          DEC     HL
F62D 22ØBF5    Ø168Ø          LD      (U),HL
F63Ø 2112ØØ    Ø169Ø          LD      HL,I8
F633 2B        Ø17ØØ          DEC     HL
```

Listing 8 continues

```
F634 E5       Ø171Ø TESTV   PUSH    HL
F635 2AØDF5   Ø172Ø         LD      HL,(V)
F638 7D       Ø173Ø         LD      A,L
F639 B4       Ø174Ø         OR      H
F63A FEØØ     Ø175Ø         CP      Ø
F63C CA46F6   Ø176Ø         JP      Z,ENDY
F63F 2B       Ø177Ø         DEC     HL
F64Ø 22ØDF5   Ø178Ø         LD      (V),HL
F643 C3Ø6F6   Ø179Ø         JP      YLOOP
F646 E1       Ø18ØØ ENDY    POP     HL
F647 C9       Ø181Ø         RET                     ;END OF AXES
F648 DDE5     Ø182Ø PLOT    PUSH    IX              ;BEGIN OF PLOT
              Ø183Ø ;THIS ROUTINE INSERTS POINT (U,V) INTO PAGE. VALUES
              Ø184Ø ;FOR U AND V MUST BE INSERTED INTO (U) AND (V) PRIOR
              Ø185Ø ;TO CALLING THIS ROUTINE.
F64A CD2CØ2   Ø186Ø         CALL    Ø22CH           ;CAUSES * TO BLINK
F64D 2AØDF5   Ø187Ø         LD      HL,(V)          ;LOAD V INTO HL
F65Ø 7C       Ø188Ø         LD      A,H             ;TO PLOT (U,V)--DIVIDE V
F651 4D       Ø189Ø         LD      C,L             ;BY DOTS,GET Q AND R THEN
F652 1EØ6     Ø19ØØ         LD      E,DOTS          ;SET BIT R IN BYTE
F654 16ØØ     Ø191Ø         LD      D,Ø             ;PAGE+Q*(PAGEW)+U
F656 CDØ1F7   Ø192Ø         CALL    DIVIDE
F659 57       Ø193Ø         LD      D,A
F65A 59       Ø194Ø         LD      E,C
F65B ED5315F5 Ø195Ø         LD      (QUOT),DE       ;SAVE Q
F65F 2217F5   Ø196Ø         LD      (RMDER),HL      ;SAVE R
F662 3A11F5   Ø197Ø         LD      A,(PAGEW)
F665 ØEØØ     Ø198Ø         LD      C,Ø             ;  DIVISOR IN DE
F667 ED5B15F5 Ø199Ø         LD      DE,(QUOT)       ;ON EXIT QUOTIENT IN AD
F66B CD17F7   Ø2ØØØ         CALL    MULT            ;  REMAINDER IN HL
F66E ED4BØBF5 Ø2Ø1Ø         LD      BC,(U)
F672 Ø9       Ø2Ø2Ø         ADD     HL,BC
F673 Ø1ØØDØ   Ø2Ø3Ø         LD      BC,PAGE
F676 Ø9       Ø2Ø4Ø         ADD     HL,BC           ;ADD ADDR OF PAGE
F677 2213F5   Ø2Ø5Ø         LD      (TEMP),HL
F67A DD2A13F5 Ø2Ø6Ø         LD      IX,(TEMP)
F67E 21Ø6ØØ   Ø2Ø7Ø         LD      HL,DOTS         ;DOTS-1 =# PRINT NEEDLES
F681 2B       Ø2Ø8Ø         DEC     HL
F682 ED5B17F5 Ø2Ø9Ø         LD      DE,(RMDER)      ;5-(RMDER) DETERMINES
F686 A7       Ø21ØØ         AND     A               ;WHICH BIT TO SET
F687 ED52     Ø211Ø         SBC     HL,DE
F689 16Ø1     Ø212Ø         LD      D,1
F68B 7D       Ø213Ø         LD      A,L
F68C B7       Ø214Ø         OR      A
F68D 28Ø5     Ø215Ø         JR      Z,RDONE         ;GO NO MORE ROTATES
F68F 47       Ø216Ø         LD      B,A
F69Ø CB22     Ø217Ø RLOOP   SLA     D               ;ROTATE TO SET BIT
F692 1ØFC     Ø218Ø         DJNZ    RLOOP
F694 DD7EØØ   Ø219Ø RDONE   LD      A,(IX)          ;GET CONTENTS OF BYTE
F697 B2       Ø22ØØ         OR      D               ;INSERT NEW INFO IN BYTE
F698 DD77ØØ   Ø221Ø         LD      (IX),A          ;STORE BACK INTO BYTE
F69B DDE1     Ø222Ø         POP     IX
F69D C9       Ø223Ø         RET                     ;END OF PLOT
F69E 3EØ3     Ø224Ø PRINT   LD      A,3             ;PUT 445 INTO GRAPHICS
F6AØ CDF6F6   Ø225Ø         CALL    OUTCHR          ;SEND 3
F6A3 ØEØØ     Ø226Ø         LD      C,Ø
F6A5 ED5BØ3F5 Ø227Ø         LD      DE,(PHIGH)
F6A9 CD17F7   Ø228Ø         CALL    MULT
F6AC 2B       Ø229Ø PLOOP   DEC     HL              ;COUNTER VERTICALLY IS
F6AD 22ØBF5   Ø23ØØ         LD      (U),HL          ;(U)=3*(PHIGH)-1
F6BØ EB       Ø231Ø         EX      DE,HL
F6B1 2A11F5   Ø232Ø         LD      HL,(PAGEW)
F6B4 7D       Ø233Ø         LD      A,L
F6B5 4C       Ø234Ø         LD      C,H
F6B6 CD17F7   Ø235Ø         CALL    MULT            ;(HL)=U*(PAGEW)
F6B9 11ØØDØ   Ø236Ø         LD      DE,PAGE
```

Listing 8 continues

gram created by the W command. This gives maximum memory during execution since the system loads the program into memory on top of the compiler and monitor sections of Pascal-80.

To get a rough idea of the memory requirements of a Pascal-80 program, consider the program Liner (Program Listing 9). This program has approximately 70 lines of text, not counting the blank lines used for cosmetic spacing. The text of Liner is from 40,704–43,362

Listing 8 continued

```
F6BC 19          Ø237Ø          ADD     HL,DE            ;(HL)=PAGE+U*(PAGEW)
F6BD 11ØØØØ      Ø238Ø          LD      DE,Ø             ;IS THE COUNTER ACROSS
F6CØ 7E          Ø239Ø GETCHR   LD      A,(HL)
F6C1 FEØ3        Ø24ØØ          CP      3                ;IF CHAR=3, SEND IT TWICE
F6C3 CCF6F6      Ø241Ø          CALL    Z,OUTCHR         ;SEND CHARACTER
F6C6 CDF6F6      Ø242Ø          CALL    OUTCHR           ;SEND CHARACTER
F6C9 13          Ø243Ø          INC     DE               ;INCREASE POINTER
F6CA 23          Ø244Ø          INC     HL               ;INCREASE POINTER
F6CB E5          Ø245Ø          PUSH    HL               ;SAVE POINTER
F6CC 2A11F5      Ø246Ø          LD      HL,(PAGEW)       ;CHECK-SEE IF LINE DONE
F6CF 2B          Ø247Ø          DEC     HL
F6DØ A7          Ø248Ø          AND     A
F6D1 ED52        Ø249Ø          SBC     HL,DE
F6D3 7C          Ø25ØØ          LD      A,H
F6D4 B5          Ø251Ø          OR      L
F6D5 E1          Ø252Ø          POP     HL
F6D6 C2CØF6      Ø253Ø          JP      NZ,GETCHR        ;CONTINUE IF NOT DONE
F6D9 3EØ3        Ø254Ø          LD      A,3              ;SEND END OF LINE SIGNAL
F6DB CDF6F6      Ø255Ø          CALL    OUTCHR
F6DE 3EØB        Ø256Ø          LD      A,ØBH
F6EØ CDF6F6      Ø257Ø          CALL    OUTCHR
F6E3 2AØBF5      Ø258Ø          LD      HL,(U)           ;CHECK-SEE IF PAGE DONE
F6E6 7C          Ø259Ø          LD      A,H
F6E7 B5          Ø26ØØ          OR      L
F6E8 C2ACF6      Ø261Ø          JP      NZ,PLOOP         ;CONTINUE IF NOT DONE
F6EB 3EØ3        Ø262Ø DONE     LD      A,3              ;SIGNAL TO EXIT GRAPHICS
F6ED CDF6F6      Ø263Ø          CALL    OUTCHR
F6FØ 3EØ2        Ø264Ø          LD      A,2
F6F2 CDF6F6      Ø265Ø          CALL    OUTCHR
F6F5 C9          Ø266Ø          RET                      ;RET TO CALLING PROG.
F6F6 F5          Ø267Ø OUTCHR   PUSH    AF               ;SUBR TO SEND CHAR
F6F7 CDD1Ø5      Ø268Ø CHCKP    CALL    5D1H             ;CHECK STATUS OF PRINTER
F6FA 2ØFB        Ø269Ø          JR      NZ,CHCKP         ;WAIT IF NOT READY
F6FC F1          Ø27ØØ          POP     AF
F6FD 32E837      Ø271Ø          LD      (37E8H),A        ;SEND IT
F7ØØ C9          Ø272Ø          RET                      ;RET TO PRINT
F7Ø1 21ØØØØ      Ø273Ø DIVIDE   LD      HL,Ø             ;FROM ZAKS
F7Ø4 Ø61Ø        Ø274Ø          LD      B,16             ;ON ENTRY DIVIDEND IN AC
F7Ø6 CB11        Ø275Ø DLOOP    RL      C                ;         DIVISOR IN DE
F7Ø8 17          Ø276Ø          RLA                      ;ON EXIT QUOTIENT IN AC
F7Ø9 ED6A        Ø277Ø          ADC     HL,HL            ;        REMAINDER IN HL
F7ØB ED52        Ø278Ø          SBC     HL,DE
F7ØD 3ØØ1        Ø279Ø          JR      NC,$+3
F7ØF 19          Ø28ØØ          ADD     HL,DE
F71Ø 3F          Ø281Ø          CCF
F711 1ØF3        Ø282Ø          DJNZ    DLOOP
F713 CB11        Ø283Ø          RL      C
F715 17          Ø284Ø          RLA
F716 C9          Ø285Ø          RET                      ;END OF DIVIDE SUBROUTINE
F717 Ø61Ø        Ø286Ø MULT     LD      B,16             ;FROM ZAKS
F719 21ØØØØ      Ø287Ø          LD      HL,Ø             ;ON ENTRY (A)=LSB MULTCND
F71C CB39        Ø288Ø MLOOP    SRL     C                ;          (C)=MSB MULTCND
F71E 1F          Ø289Ø          RRA                      ;          (DE)=MULTIPLIER
F71F 3ØØ1        Ø29ØØ          JR      NC,MLOOP1        ;ON EXIT (HL)=PRODUCT
F721 19          Ø291Ø          ADD     HL,DE
F722 EB          Ø292Ø MLOOP1   EX      DE,HL
F723 29          Ø293Ø          ADD     HL,HL
F724 EB          Ø294Ø          EX      DE,HL
F725 1ØF5        Ø295Ø          DJNZ    MLOOP
F727 C9          Ø296Ø          RET                 ;END OF MULT SUBROUTINE
ØØØØ             Ø297Ø          END
ØØØØØ TOTAL ERRORS
```

```
ØØØØ PROGRAM LINER;
ØØØ3 CONST TIGER=-2816;ADRU=-28Ø5;ADRV=-28Ø3;
ØØØC I6=16;I8=18;
ØØØC VAR I,J,U,V,Z,WIDTH,HEIGHT,HØ,KØ,INCREMENT:INTEGER;
ØØ34     H1,K1,X,Y:REAL;
ØØ44     ANSWER:CHAR;
ØØ48
ØØ48 PROC INIT(VAR PWIDE,PHIGH,HØ,KØ:INTEGER);
ØØ61 CONST ADRHØ=-28Ø9;ADRKØ=-28Ø7;ADRPHIGH=-2813;ADRPWIDE=-2811;
ØØ62 BEGIN
ØØ67   WRITELN('HOW MANY UNITS FOR X-AXIS AND FOR Y-AXIS');
ØØ9C   READ(PWIDE,PHIGH);
ØØA9   POKE(ADRPHIGH,PHIGH); POKE(ADRPHIGH+1,Ø);
ØØBC   POKE(ADRPWIDE,PWIDE); POKE(ADRPWIDE+1,Ø);
ØØCF   WRITELN('ENTER HØ AND KØ TO PLACE THE ORIGIN');
ØØFF   READ(HØ,KØ);
Ø1ØC   POKE(ADRHØ,HØ); POKE(ADRHØ+1,Ø);
Ø11F   POKE(ADRKØ,KØ); POKE(ADRKØ+1,Ø)
Ø132 END;(*INIT*)
Ø133
Ø133 FUNC MIN(X,Y:INTEGER):INTEGER;
Ø13F BEGIN
Ø145   IF X<Y THEN MIN:=X ELSE MIN:=Y
Ø169 END;(*MIN*)
Ø16C
Ø16C FUNC MAX(X,Y:INTEGER):INTEGER;
Ø178 BEGIN
Ø17E   IF X<Y THEN MAX:=Y ELSE MAX:=X
Ø1A2 END;(*MAX*)
Ø1A5
Ø1A5 PROC FILL(X,Y:INTEGER);
Ø1B1 VAR A1,A2,V:INTEGER;
Ø1BE   BEGIN
Ø1C3     A1:=MAX(MIN(X,Y),Ø)+1;
Ø1EØ     A2:=MIN(MAX(X,Y),HEIGHT)-1;
Ø2Ø1     V:=A1;
Ø2ØB     WHILE V<=A2 DO
Ø21A       BEGIN
Ø21A         POKE(ADRV,V); POKE(ADRV+1,Ø);
Ø22D         V:=V+2;
Ø23C         Z:=CALL(TIGER,2)
Ø246       END
Ø248   END;(*FILL*)
Ø24C
Ø24C PROC LINE;
Ø25Ø VAR A1,A2,U,V,OLDV:INTEGER;
Ø265     X1,Y1,X2,Y2,M,B:REAL;
Ø27D BEGIN
Ø282   WRITELN('ENTER COORDINATES OF FIRST POINT');
Ø2AF   READ(X1,Y1);
Ø2BC   WRITELN('ENTER COORDINATES OF SECOND POINT');
Ø2EA   READ(X2,Y2);
Ø2F7   WHILE (X1<>X2) OR (Y1<>Y2) DO
Ø313   BEGIN
Ø313   IF Y1=Y2 THEN INCREMENT:=2 ELSE INCREMENT:=1;
Ø335   IF (X1<>X2) THEN
Ø344     BEGIN
Ø344       A1:=MIN(ROUND(I6*(X1+HØ)),ROUND(I6*(X2+HØ)));
Ø373       A2:=MAX(ROUND(I6*(X1+HØ)),ROUND(I6*(X2+HØ)));
Ø3A2       M:=(Y1-Y2)/(X1-X2);
Ø3C1       B:=(X1*Y2-X2*Y1)/(X1-X2);
Ø3EE       U:=A1;
Ø3F8       WHILE U<=A2 DO
Ø4Ø7         BEGIN
Ø4Ø7           V:=ROUND(I8*(KØ+M*(U/I6-HØ)+B));
Ø439           IF (V>=Ø) AND (V<=HEIGHT) THEN
```

Listing 9 continues

Listing 9 continued

```
Ø451           BEGIN
Ø451             IF V=HEIGHT THEN V:=HEIGHT-1;
Ø46F             POKE(ADRU,U); POKE(ADRU+1,Ø);
Ø482             POKE(ADRV,V); POKE(ADRV+1,Ø);
Ø495             Z:=CALL(TIGER,2)
Ø49F           END;
Ø4A1           IF(U =A1) THEN OLDV:=V;
Ø4BA           IF(ABS(V-OLDV)>3) THEN FILL(V,OLDV);
Ø4DD           U:=U+INCREMENT;
Ø4EE           OLDV:=V
Ø4F6         END
Ø4F8     END
Ø4FB   ELSE
Ø4FE     BEGIN
Ø4FE       U:=ROUND(I6*(X1+HØ));
Ø516       IF U=WIDTH THEN U:=U-2;
Ø534       POKE(ADRU,U); POKE(ADRU+1,Ø);
Ø547       FILL(ROUND(I8*(Y1+KØ)), ROUND(I8*(Y2+KØ)))
Ø57Ø     END;
Ø57Ø   X1:=X2;Y1:=Y2;
Ø584   WRITELN('ENTER NEXT POINT--TO END THIS SEQUENCE OF LINES');
Ø5CØ   WRITELN('ENTER THE SAME POINT TWICE');
Ø5E7   READ(X2,Y2)
Ø5F4   END
Ø5F4 END;(*LINE*)
Ø5F8
Ø5F8 BEGIN
Ø5F8   WRITELN('THIS PROGRAM ALLOWS YOU TO DRAW A SEQUENCE OF');
Ø632   WRITELN('LINES. TO END A SEQUENCE, ENTER THE SAME POINT');
Ø66D   WRITELN('TWICE. THEN YOU HAVE THE OPTION OF STARTING A');
Ø6A7   WRITELN('NEW SEQUENCE OF LINES OR PRINTING THE GRAPH');
Ø6DF   INIT(WIDTH,HEIGHT,HØ,KØ);
Ø6EE   WIDTH:=I6*WIDTH; HEIGHT:=I8*HEIGHT;
Ø7ØC   Z:=CALL(TIGER,Ø);
Ø716   WRITELN('DO YOU WANT AXES DRAWN (Y OR N)');
Ø742   REPEAT ANSWER:=INKEY UNTIL ANSWER<>CHR(Ø);
Ø756   IF ANSWER = 'Y'   THEN Z:=CALL(TIGER,1);
Ø76F   LINE;
Ø772   REPEAT
Ø772     BEGIN
Ø772       WRITELN('DO YOU WANT TO DO ANOTHER SEQUENCE OF LINES?');
Ø7AB       WRITELN('(Y OR N)--N PRINTS THE PAGE AND ENDS THIS');
Ø7E1       WRITELN('PROGRAM.');
Ø7F6       REPEAT ANSWER:=INKEY UNTIL ANSWER<>CHR(Ø);
Ø8ØA       IF ANSWER='Y' THEN LINE;
Ø81A     END;
Ø81A   UNTIL ANSWER='N';
Ø827   Z:=CALL(TIGER,3)
Ø831 END.
```

Program Listing 10

```
ØØØØ PROGRAM SCALER;
ØØØ3 CONST TIGER=-2816;ADRU=-28Ø5;ADRV=-28Ø3;
ØØØC I6=16;I8=18;
ØØØC TYPE POINT=ARRAY(.1..2.) OF REAL;
ØØ12 VAR I,J,U,V,Z,WIDTH,HEIGHT,HØ,KØ,OLDV:INTEGER;
ØØ3A     XSCALER,YSCALER:REAL;
ØØ42     ANSWER:CHAR;
ØØ46
ØØ46 PROC INIT(VAR PWIDE,PHIGH,HØ,KØ:INTEGER);
ØØ5F CONST ADRHØ=-28Ø9;ADRKØ=-28Ø7;ADRPHIGH=-2813;ADRPWIDE=-2811;
ØØ6Ø BEGIN
```

Listing 10 continues

```
0065   WRITELN('HOW MANY UNITS FOR X-AXIS AND FOR Y-AXIS');
009A   READ(PWIDE,PHIGH);
00A7   POKE(ADRPHIGH,PHIGH); POKE(ADRPHIGH+1,0);
00BA   POKE(ADRPWIDE,PWIDE); POKE(ADRPWIDE+1,0);
00CD   WRITELN('ENTER H0 AND K0 TO PLACE THE ORIGIN');
00FD   READ(H0,K0);
010A   POKE(ADRH0,H0); POKE(ADRH0+1,0);
011D   POKE(ADRK0,K0); POKE(ADRK0+1,0)
0130 END;(*INIT*)
0131
0131 FUNC MIN(X,Y:INTEGER):INTEGER;
013D BEGIN
0143   IF X<Y THEN MIN:=X ELSE MIN:=Y
0167 END;(*MIN*)
016A
016A FUNC MAX(X,Y:INTEGER):INTEGER;
0176 BEGIN
017C   IF X<Y THEN MAX:=Y ELSE MAX:=X
01A0 END;(*MAX*)
01A3
01A3 PROC FILL(X,Y:INTEGER);
01AF VAR A1,A2,V:INTEGER;
01BC   BEGIN
01C1     A1:=MAX(MIN(X,Y),0)+1;
01DE     A2:=MIN(MAX(X,Y),HEIGHT)-1;
01FF     V:=A1;
0209     WHILE V<=A2 DO
0218       BEGIN
0218         POKE(ADRV,V); POKE(ADRV+1,0);
022B         V:=V+2;
023A         Z:=CALL(TIGER,2)
0244       END
0246   END;(*FILL*)
024A
024A FUNC F(X:REAL):REAL;
0252   BEGIN  F:=(X-6)*X*(X+10)
0276 END;(*FUNC F*)
027B
027B
027B BEGIN
027B   INIT(WIDTH,HEIGHT,H0,K0);
028A   WIDTH:=I6*WIDTH; HEIGHT:=I8*HEIGHT;
02A8   Z:=CALL(TIGER,0);
02B2   WRITELN('DO YOU WANT AXES DRAWN (Y OR N)');
02DE   REPEAT ANSWER:=INKEY UNTIL ANSWER<>CHR(0);
02F2   IF ANSWER = 'Y'   THEN Z:=CALL(TIGER,1);
030B   WRITELN('ENTER THE SCALE FACTOR FOR X-AXIS');
0339   READ(XSCALER);
0340   WRITELN('ENTER THE SCALE FACTOR FOR Y-AXIS');
036E   READ(YSCALER);
0375
0375   FOR U:=0 TO WIDTH DO
0389      BEGIN
0389       V:=ROUND(18*(K0+(1/YSCALER)*F(XSCALER*(U/16-H0))));
03C4       IF (V>=0) AND (V<=HEIGHT) THEN
03DC         BEGIN
03DC           POKE(ADRU,U);POKE(ADRU+1,0);
03EF           POKE(ADRV,V);POKE(ADRV+1,0);
0402           Z:=CALL(TIGER,2)
040C         END;
040E       IF U=0 THEN OLDV:=V;
0423       IF ABS(V-OLDV)>2 THEN FILL(V,OLDV);
0446       OLDV:=V
044E     END;
0453   Z:=CALL(TIGER,3)
045D END.
```

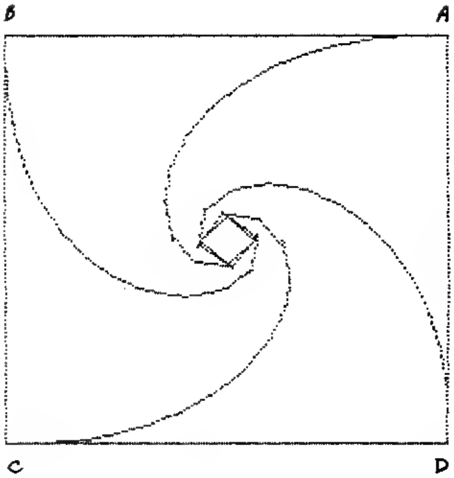


Figure 23

(9F00H–A962H), and its compiled version is from 43,520–45,290 (AA00H–B0EAH). This leaves the memory area from 45,290 to 53,248 free. All the programs in this article fit within these limits and it is not necessary to use the W and X commands.

Pascal-80 Programs for Graphing

In the following Pascal-80 programs, certain procedures and functions are equivalent to segments of the Basic programs in Part I. I have tried, as much as possible, to keep the same names for the variables as in the Basic programs. Liner incorporates the functions and procedures common to the Pascal-80 programs that call Tigerpas. These procedures are described below.

Procedure INIT initializes the variables. HO, KO, PHIGH, and PWIDE have the same meaning as in the Basic programs in Part I, namely for the placement of the x-y coordinate axes and for the number of units for the lengths of the x and y axes. Their values are entered using a read statement and the four values are POKEd into the appropriate memory locations. The addresses for these variables are defined in the CONST declaration statement.

Functions MIN and MAX each have two arguments X and Y, and return the minimum and maximum of X and Y, respectively.

Procedure Fill is similar to the subroutine in the Basic programs. It is used to fill the vertical gaps in graphs that are rising or falling rapidly. The statements defining the values A1 and A2 find the vertical range of the v values to be filled and keep their values within the boundaries of the page. For example, in finding the lower limit A1, first find the minimum of X = V and Y = OLDV. To keep the v values nonnegative, take the maximum of the MIN(X,Y) and 0. Similarly, the upper limit A2 is found. The Pascal-80 function Round finds the integer nearest its argument. The equivalent statement in the Basic programs is INT(argument + .5). Once A1 and A2 have been determined, the rest of the procedure sends the appropriate values to the Plot subroutine.

Program Listing 11

```
0000 PROGRAM DOGCAT;
0003 CONST TIGER=-2816;ADRU=-2805;ADRV=-2803;
000C I6=16;I8=18;
000C TYPE POINTS=ARRAY(.1..2.) OF REAL;
0012 VAR I,J,U,V,Z,PWIDE,PHIGH ,H0,K0,STOPDOG,STOPCAT:INTEGER;
003E     WIDTH,HEIGHT:INTEGER;
0046     H1,K1,X1,Y1,X2,Y2,K,DOGVELOCITY,CATVELOCITY:REAL;
006A     DOG,CAT,TREE:POINTS;
0076     ANSWER:CHAR;
007A
007A PROC INIT;
0083 CONST ADRH0=-2809;ADRK0=-2807;ADRPHIGH=-2813;ADRPWIDE=-2811;
0084 BEGIN
0089   PWIDE:=14; PHIGH:=9; H0:=1; K0:=1;
00A9   POKE(ADRPHIGH,PHIGH); POKE(ADRPHIGH+1,0);
00BC   POKE(ADRPWIDE,PWIDE); POKE(ADRPWIDE+1,0);
00CF   POKE(ADRH0,H0); POKE(ADRH0+1,0);
00E2   POKE(ADRK0,K0); POKE(ADRK0+1,0)
00F5 END;(*INIT*)
00F6
00F6 FUNC MIN(X,Y:INTEGER):INTEGER;
0102 BEGIN
0108   IF X<Y THEN MIN:=X ELSE MIN:=Y
012C END;(*MIN*)
012F
012F FUNC MAX(X,Y:INTEGER):INTEGER;
013B BEGIN
0141   IF X<Y THEN MAX:=Y ELSE MAX:=X
0165 END;(*MAX*)
0168
0168 PROC FILL(X,Y:INTEGER);
0174 VAR A1,A2,V:INTEGER;
```

Listing 11 continues

Listing 11 continued

```
Ø181    BEGIN
Ø186      A1:=MAX(MIN(X,Y),Ø)+1;
Ø1A3      A2:=MIN(MAX(X,Y),HEIGHT)-1;
Ø1C4      V:=A1;
Ø1CE      WHILE V<=A2 DO
Ø1DD        BEGIN
Ø1DD          POKE(ADRV,V); POKE(ADRV+1,Ø);
Ø1FØ          V:=V+2;
Ø1FF          Z:=CALL(TIGER,2)
Ø2Ø9        END
Ø2ØB    END;(*FILL*)
Ø2ØF
Ø2ØF PROC LINE;
Ø213 VAR A1,A2,U,V,OLDV,I:INTEGER;
Ø22C     M,B:REAL;
Ø234 BEGIN
Ø239   IF (X1<>X2) THEN
Ø248     BEGIN
Ø248       A1:=MIN(ROUND(I6*(X1+HØ)),ROUND(I6*(X2+HØ)));
Ø277       A2:=MAX(ROUND(I6*(X1+HØ)),ROUND(I6*(X2+HØ)));
Ø2A6       M:=(Y1-Y2)/(X1-X2);
Ø2C5       B:=(X1*Y2-X2*Y1)/(X1-X2);
Ø2F2       FOR U:=A1 TO A2 DO
Ø3ØA         BEGIN
Ø3ØA           V:=ROUND(I8*(KØ+M*(U/I6-HØ)+B));
Ø33C           IF (V>=Ø) AND (V<=HEIGHT) THEN
Ø354           BEGIN
Ø354             IF V=HEIGHT THEN V:=HEIGHT-1;
Ø372             POKE(ADRU,U); POKE(ADRU+1,Ø);
Ø385             POKE(ADRV,V); POKE(ADRV+1,Ø);
Ø398             Z:=CALL(TIGER,2)
Ø3A2           END;
Ø3A4           IF(U =A1) THEN OLDV:=V;
Ø3BD           IF(ABS(V-OLDV)>2) THEN FILL(V,OLDV);
Ø3EØ           OLDV:=V
Ø3E8         END
Ø3EA     END
Ø3ED   ELSE
Ø3FØ     BEGIN
Ø3FØ       U:=ROUND(I6*(X1+HØ));
Ø4Ø8       IF U=WIDTH THEN U:=U-2;
Ø426       POKE(ADRU,U); POKE(ADRU+1,Ø);
Ø439       FILL(ROUND(I8*(Y1+KØ)), ROUND(I8*(Y2+KØ)))
Ø462     END
Ø462 END;(*LINE*)
Ø463
Ø463 PROC MOVE(K:REAL;VAR X,Y:POINTS);
Ø473 VAR M,P,Q,DISTANCE:REAL;
Ø484 BEGIN
Ø489   P:=Y(.1.)-X(.1.); Q:=Y(.2.)-X(.2.);
Ø4CB   DISTANCE:=SQRT(SQR(P)+SQR(Q));
Ø4E2   IF (DISTANCE <= K) THEN
Ø4F1     BEGIN
Ø4F1       X1:=Y(.1.); Y1:=Y(.2.);
Ø515     END
Ø515   ELSE
Ø518     BEGIN
```

Listing 11 continues

Listing 11 continued

```
Ø518        M:=K/DISTANCE;
Ø529        X1:=X(.1.) + M*P; Y1:=X(.2.) + M*Q;
Ø569      END;
Ø569    X2:=X(.1.); Y2:=X(.2.);
Ø58D    LINE;
Ø59Ø    X(.1.):=X1; X(.2.):=Y1
Ø5B2  END;(*MOVE*)
Ø5B5
Ø5B5  BEGIN
Ø5B5    INIT;
Ø5B8    WIDTH:=I6*PWIDE; HEIGHT:=I8*PHIGH;
Ø5D6    Z:=CALL(TIGER,Ø);
Ø5EØ    WRITELN('DO YOU WANT AXES DRAWN (Y OR N)');
Ø6ØC    REPEAT ANSWER:=INKEY UNTIL ANSWER<>CHR(Ø);
Ø62Ø    IF ANSWER = 'Y'   THEN Z:=CALL(TIGER,1);
Ø639    DOG(.1.):=13; DOG(.2.):=Ø;
Ø659    CAT(.1.):=Ø; CAT(.2.):=Ø;
Ø677    TREE(.1.):=Ø; TREE(.2.):=8;
Ø697    DOGVELOCITY:=2.Ø;
Ø6A1    CATVELOCITY:=1.5;
Ø6AB    STOPDOG:=Ø; STOPCAT:=Ø;
Ø6B7    REPEAT
Ø6B7      IF((CAT(.1.)=TREE(.1.)) AND (CAT(.2.)=TREE(.2.)))
Ø6FØ        THEN STOPCAT:=1
Ø6F9      ELSE MOVE(DOGVELOCITY,DOG,CAT);
Ø7ØC      IF(STOPCAT=1)OR((DOG(.1.)=CAT(.1.))AND(DOG(.2.)=CAT(.2.)))
Ø74F        THEN STOPDOG:=1
Ø759      ELSE MOVE(CATVELOCITY,CAT,TREE);
Ø76C    UNTIL (STOPDOG=1) OR (STOPCAT=1);
Ø784    Z:=CALL(TIGER,3);
Ø79Ø    WRITELN(LP);
Ø797    IF (STOPCAT<>1) THEN
Ø7A4      WRITELN(LP,'DOG GOT CAT AT  (',DOG(.1.):3,',',
Ø7D3        DOG(.2.):3,')')
Ø7EB    ELSE  WRITELN(LP,'CAT GOT TO THE TREE');
Ø815    WRITELN(LP,'VELOCITY OF DOG: ',DOGVELOCITY,
Ø833      '  VELOCITY OF CAT: ',CATVELOCITY);
Ø85A    WRITELN(LP,'ORIGIN AT (',HØ,',',KØ,')');
Ø88C    WRITELN(LP,'PHIGH: ',PHIGH,'   PWIDE: ',PWIDE)
Ø8B9  END.
```

Procedure Line is designed to draw a sequence of connected line segments. That is, given a sequence of points P1, P2, P3, ... Line draws the line segments P1 to P2, P2 to P3, and so forth. The procedure begins by asking for the first two points, and the line segment joining these two points is inserted into the page. After the line segment is drawn, the user is asked for the next point in the sequence. If the point entered is the same as the last point entered, the procedure ends; otherwise the While loop continues and the next line segment is drawn.

Vertical lines and nonvertical lines must be done in different cases, just as in the Basic programs in Part I. The variable INCREMENT is assigned a value of one or two, according to whether or not the line segment is vertical or not vertical, respectively. INCREMENT is used to space the dots of the graph, and without this change in INCREMENT, vertical lines are darker when graphed. In addition, it saves time by reducing the number of computations and POKEs to be performed.

When control returns to the body of the program, the user has the option of beginning a new line-segment sequence. Depending upon the answer, the program either ends or uses the Line procedure again. Figure 18 was drawn using Liner.

Change the Scale of a Graph

There comes a time in the graphing of functions when the implicit restriction on the domains and ranges of functions to be graphed must be overcome. For example, if we want to graph the national debt over the past 50 years, obviously the graph must be scaled to fit within the dimensions of the maximum size page permitted. The program Scaler (Program Listing 10) is an example of graphing a function for x's in the range from −20 to 20 and y's ranging from −400 to 400, within the dimensions of a 2-by-2 page. In the previous

```
ØØØØ PROGRAM BUGS;
ØØØ3 CONST TIGER=-2816;ADRU=-28Ø5;ADRV=-28Ø3;
ØØØC I6=16;I8=18;
ØØØC TYPE POINTS=ARRAY(.1..2.) OF REAL;
ØØ12 VAR I,J,U,V,Z,WIDTH,HEIGHT,HØ,KØ:INTEGER;
ØØ36     H1,K1,X1,Y1,X2,Y2,K:REAL;
ØØ52     A,B,C,D,OLDA:POINTS;
ØØ66     ANSWER:CHAR;
ØØ6A
ØØ6A PROC INIT(VAR PWIDE,PHIGH,HØ,KØ:INTEGER);
ØØ83 CONST ADRHØ=-28Ø9;ADRKØ=-28Ø7;ADRPHIGH=-2813;ADRPWIDE=-2811;
ØØ84 BEGIN
ØØ89   PWIDE:=14; PHIGH:=14; HØ:=7; KØ:=7;
ØØA9   POKE(ADRPHIGH,PHIGH); POKE(ADRPHIGH+1,Ø);
ØØBC   POKE(ADRPWIDE,PWIDE); POKE(ADRPWIDE+1,Ø);
ØØCF   POKE(ADRHØ,HØ); POKE(ADRHØ+1,Ø);
ØØE2   POKE(ADRKØ,KØ); POKE(ADRKØ+1,Ø)
ØØF5 END;(*INIT*)
ØØF6
ØØF6 FUNC MIN(X,Y:INTEGER):INTEGER;
Ø1Ø2 BEGIN
Ø1Ø8   IF X<Y THEN MIN:=X ELSE MIN:=Y
Ø12C END;(*MIN*)
Ø12F
Ø12F FUNC MAX(X,Y:INTEGER):INTEGER;
Ø13B BEGIN
Ø141   IF X<Y THEN MAX:=Y ELSE MAX:=X
Ø165 END;(*MAX*)
Ø168
Ø168 PROC FILL(X,Y:INTEGER);
Ø174 VAR A1,A2,V:INTEGER;
Ø181   BEGIN
Ø186     A1:=MAX(MIN(X,Y),Ø)+1;
Ø1A3     A2:=MIN(MAX(X,Y),HEIGHT)-1;
Ø1C4     V:=A1;
Ø1CE     WHILE V<=A2 DO
Ø1DD       BEGIN
Ø1DD         POKE(ADRV,V); POKE(ADRV+1,Ø);
Ø1FØ         V:=V+2;
Ø1FF         Z:=CALL(TIGER,2)
Ø2Ø9       END
Ø2ØB   END;(*FILL*)
Ø2ØF
Ø2ØF PROC LINE;
Ø213 VAR A1,A2,U,V,OLDV,I:INTEGER;
Ø22C     M,B:REAL;
Ø234 BEGIN
Ø239   IF (X1<>X2) THEN
Ø248     BEGIN
Ø248       A1:=MIN(ROUND(I6*(X1+HØ)),ROUND(I6*(X2+HØ)));
Ø277       A2:=MAX(ROUND(I6*(X1+HØ)),ROUND(I6*(X2+HØ)));
Ø2A6       M:=(Y1-Y2)/(X1-X2);
Ø2C5       B:=(X1*Y2-X2*Y1)/(X1-X2);
Ø2F2       FOR U:=A1 TO A2 DO
Ø3ØA         BEGIN
Ø3ØA           V:=ROUND(I8*(KØ+M*(U/I6-HØ)+B));
```

Listing 12 continues

Turn your computer into an advanced operating system. . .

Use Instant Software's Utilities for your TRS-80*

•Enhance •Accelerate •Expand.

SUPER UTILITY PLUS The most powerful program of its kind! Machine language, stand-alone program with its own I/O routines. No ROM or DOS calls. Works on single AND double density systems. Features ZAP, PURGE, FORMAT, DISK COPY, TAPE COPY, DISK REPAIR, MEMORY, FILE, and CONFIGURE SYSTEM functions. Very sophisticated, menu driven, single RAM resident program. A must for every serious TRS-80 disk installation.
48K RAM Model I & III Disk

DOSPLUS 3.4 The fastest, smoothest, easiest to use Disk operating system for your TRS-80! Jam-packed with powerful options: BASIC Array Sort, Tape/Disk—Disk/Tape Utility, Controlled Screen Input, Random Access and ASCII modification on Diskdump, BASIC Checks for Active "DO", Backup and Format from a "DO" file, and much much more. Energize your operating system with DOSPLUS 3.4!
TRS-80 2 for Model I, 1 for Model III Disk

KEY COMMANDER Easy editing of programs, and direct entry of graphics into Basic programs are yours with this highly sophisticated yet easy to use utility package. Features On Screen Editing, Graphic Print Statement, Assignable Keys, and Total Flexibility. Automatically relocates into high memory and protects itself in memory. Enhance the power and capabilities of your keyboard!
Model I/Model III 16K and up Tape

ABE Accelerate your writing and editing of Basic programs with the Advanced Basic Editor. Dramatically improves your Basic editing and programming capabilities by providing such indispensable features as Formatted Screen, Global Search and Replace, Line Copy Ability, Assignable Keys, Called From Basic. Advance your skills as a programmer with ABE.
Model I 32K TRSDOS 2.3 or NEWDOS 80 v2 Disk in 32K & 48K version.

Instant Software™
Peterborough, N.H. 03458 USA A subsidiary of Wayne Green Inc.
The best software under the sun.

To place your order for Instant Software Programs, call toll free
1-800-258-5473
outside US & Canada call Worldwide Media 212/686-1520

*TRS-80 is a trademark of the Radio Shack division of Tandy Corp.

ANU80

Yes I want to order:
_____ABE @ $19.95 ea.
_____Key Commander @ $29.95 ea.
_____Super Utility Plus @ $74.95 ea.
_____Dos Plus 3.4 Single Density @ $149.95 ea.
_____Dos Plus 3.4 Double Density @ $149.95 ea.
_____Dos Plus 3.4 Mod III Version @ $149.95 ea.

Shipping & Handling $2.50
For C.O.D. orders add $1.50 extra
Total of above__________

_____VISA _____MC _____AE _____Check _____Money Order

Name__________ Card #__________
Address__________ Interbank #__________
City__________ Exp. Date__________
State__________ Zip__________ Signature__________

Instant Software™
Peterborough, N.H. 03458 USA A subsidiary of Wayne Green Inc.

Call toll free 1-800-258-5473
Outside U.S. & Canada call Worldwide Media 212-686-1520

118

Listing 12 continued

```
033C            IF (V>=0) AND (V<=HEIGHT) THEN
0354            BEGIN
0354              IF V=HEIGHT THEN V:=HEIGHT-1;
0372              POKE(ADRU,U); POKE(ADRU+1,0);
0385              POKE(ADRV,V); POKE(ADRV+1,0);
0398              Z:=CALL(TIGER,2)
03A2            END;
03A4            IF(U =A1) THEN OLDV:=V;
03BD            IF(ABS(V-OLDV)>2) THEN FILL(V,OLDV);
03E0            OLDV:=V
03E8          END
03EA      END
03ED    ELSE
03F0      BEGIN
03F0        U:=ROUND(I6*(X1+H0));
0408        IF U=WIDTH THEN U:=U-2;
0426        POKE(ADRU,U); POKE(ADRU+1,0);
0439        FILL(ROUND(I8*(Y1+K0)), ROUND(I8*(Y2+K0)))
0462      END
0462 END;(*LINE*)
0463
0463 PROC MOVE(VAR X,Y:POINTS);
046F VAR M,P,Q:REAL;
047C BEGIN
0481   P:=Y(.1.)-X(.1.); Q:=Y(.2.)-X(.2.);
04C3   M:=K/SQRT(SQR(P)+SQR(Q));
04E1   X1:=X(.1.)+M*P; Y1:=X(.2.)+M*Q;
0521   X2:=X(.1.); Y2:=X(.2.);
0545   LINE;
0548   X(.1.):=X1; X(.2.):=Y1
056A END;(*MOVE*)
056D
056D PROC SQUARE;
0571 BEGIN
0577 X1:=6;Y1:=6; X2:=-6;Y2:=6;LINE; X2:=6; Y2:=-6; LINE;
05B7 X1:=-6;Y1:=-6;LINE; X2:=-6;Y2:=6;LINE
05E7 END;(*SQUARE*)
05E8
05E8 BEGIN
05E8   INIT(WIDTH,HEIGHT,H0,K0);
05F7   WIDTH:=I6*WIDTH; HEIGHT:=I8*HEIGHT;
0615   Z:=CALL(TIGER,0);
061F   WRITELN('DO YOU WANT AXES DRAWN (Y OR N)');
064B   REPEAT ANSWER:=INKEY UNTIL ANSWER<>CHR(0);
065F   IF ANSWER = 'Y'   THEN Z:=CALL(TIGER,1);
0678   SQUARE;
067B   K:=1.0;
0685   A(.1.):=6 ;A(.2.):=6;
06A7   B(.1.):=-6 ;B(.2.):=6;
06CB   C(.1.):=-6; C(.2.):=-6;
06F1   D(.1.):=6;  D(.2.):=-6;
0715   OLDA(.1.):=A(.1.); OLDA(.2.):=A(.2.);
0749   FOR I:=0 TO 40 DO
075B   BEGIN
075B     MOVE(A,B); MOVE(B,C); MOVE(C,D);MOVE(D,OLDA);
077F     OLDA(.1.):=A(.1.); OLDA(.2.):=A(.2.)
07B1   END;
```

Listing 12 continues

Listing 12 continued

```
07B6    Z:=CALL(TIGER,3)
07C0 END.
```

Program Listing 13

```
0000 PROGRAM SOLUTION;
0003 CONST TIGER=-2816;ADRU=-2805;ADRV=-2803;
000C I6=16;I8=18;
000C VAR I,J,U,V,Z,WIDTH,HEIGHT,H0,K0:INTEGER;
0030     X1,Y1,X2,Y2,R,ALPHA,BETA:REAL;
004C     ANSWER:CHAR;
0050
0050 PROC INIT(VAR PWIDE,PHIGH,H0,K0:INTEGER);
0069 CONST ADRH0=-2809;ADRK0=-2807;ADRPHIGH=-2813;ADRPWIDE=-2811;
006A BEGIN
006F   PWIDE:=14; PHIGH:=14; H0:=7; K0:=7;
008F   POKE(ADRPHIGH,PHIGH); POKE(ADRPHIGH+1,0);
00A2   POKE(ADRPWIDE,PWIDE); POKE(ADRPWIDE+1,0);
00B5   POKE(ADRH0,H0); POKE(ADRH0+1,0);
00C8   POKE(ADRK0,K0); POKE(ADRK0+1,0)
00DB END;(*INIT*)
00DC
00DC FUNC MIN(X,Y:INTEGER):INTEGER;
00E8 BEGIN
00EE   IF X<Y THEN MIN:=X ELSE MIN:=Y
0112 END;(*MIN*)
0115
0115 FUNC MAX(X,Y:INTEGER):INTEGER;
0121 BEGIN
0127   IF X<Y THEN MAX:=Y ELSE MAX:=X
014B END;(*MAX*)
014E
014E PROC FILL(X,Y:INTEGER);
015A VAR A1,A2,V:INTEGER;
0167   BEGIN
016C     A1:=MAX(MIN(X,Y),0)+1;
0189     A2:=MIN(MAX(X,Y),HEIGHT)-1;
01AA     V:=A1;
01B4     WHILE V<=A2 DO
01C3       BEGIN
01C3         POKE(ADRV,V); POKE(ADRV+1,0);
01D6         V:=V+2;
01E5         Z:=CALL(TIGER,2)
01EF       END
01F1   END;(*FILL*)
01F5
01F5 PROC LINE;
01F9 VAR A1,A2,U,V,OLDV,I:INTEGER;
0212     M,B:REAL;
021A BEGIN
021F   IF (X1<>X2) THEN
022E     BEGIN
022E       A1:=MIN(ROUND(I6*(X1+H0)),ROUND(I6*(X2+H0)));
025D       A2:=MAX(ROUND(I6*(X1+H0)),ROUND(I6*(X2+H0)));
028C       M:=(Y1-Y2)/(X1-X2);
```

Listing 13 continues

Listing 13 continues

```
Ø2AB        B:=(X1*Y2-X2*Y1)/(X1-X2);
Ø2D8        FOR U:=A1 TO A2 DO
Ø2FØ          BEGIN
Ø2FØ            V:=ROUND(I8*(KØ+M*(U/I6-HØ)+B));
Ø322            IF (V>=Ø) AND (V<=HEIGHT) THEN
Ø33A            BEGIN
Ø33A              IF V=HEIGHT THEN V:=HEIGHT-1;
Ø358              POKE(ADRU,U); POKE(ADRU+1,Ø);
Ø36B              POKE(ADRV,V); POKE(ADRV+1,Ø);
Ø37E              Z:=CALL(TIGER,2)
Ø388            END;
Ø38A            IF(U =A1) THEN OLDV:=V;
Ø3A3            IF(ABS(V-OLDV)>2) THEN FILL(V,OLDV);
Ø3C6            OLDV:=V
Ø3CE          END
Ø3DØ      END
Ø3D3    ELSE
Ø3D6      BEGIN
Ø3D6        U:=ROUND(I6*(X1+HØ));
Ø3EE        IF U=WIDTH THEN U:=U-2;
Ø4ØC        POKE(ADRU,U); POKE(ADRU+1,Ø);
Ø41F        FILL(ROUND(I8*(Y1+KØ)), ROUND(I8*(Y2+KØ)))
Ø448      END
Ø448 END;(*LINE*)
Ø449
Ø449 PROC SQUARE;
Ø44D BEGIN
Ø453 X1:=6;Y1:=6; X2:=-6;Y2:=6;LINE; X2:=6; Y2:=-6; LINE;
Ø493 X1:=-6;Y1:=-6;LINE; X2:=-6;Y2:=6;LINE
Ø4C3 END;(*SQUARE*)
Ø4C4
Ø4C4 BEGIN
Ø4C4   INIT(WIDTH,HEIGHT,HØ,KØ);
Ø4D3   WIDTH:=I6*WIDTH; HEIGHT:=I8*HEIGHT;
Ø4F1   Z:=CALL(TIGER,Ø);
Ø4FB   WRITELN('DO YOU WANT AXES DRAWN (Y OR N)');
Ø527   REPEAT ANSWER:=INKEY UNTIL ANSWER<>CHR(Ø);
Ø53B   IF ANSWER = 'Y'   THEN Z:=CALL(TIGER,1);
Ø554   SQUARE;
Ø557   POKE(ADRU+1,Ø);POKE(ADRV+1,Ø);
Ø56B   ALPHA:=PI/4;
Ø57E   BETA:=6*SQRT(2);
Ø58E   WHILE ALPHA<3*PI/2 DO
Ø5AB     BEGIN
Ø5AB       R:=BETA*EXP(PI/4-ALPHA);
Ø5DØ       U:=ROUND(I6*(R*COS(ALPHA) + HØ));
Ø5F3       V:=ROUND(I8*(R*SIN(ALPHA) + KØ));
Ø616       IF (V>=Ø) AND (V<=HEIGHT) AND (U>=Ø) AND (U<=WIDTH) THEN
Ø644         BEGIN
Ø644           POKE(ADRU,U);
Ø64D           POKE(ADRV,V);
Ø656           Z:=CALL(TIGER,2)
Ø66Ø         END;
Ø662       ALPHA:=ALPHA+Ø.Ø1
Ø66F     END;
Ø676   Z:=CALL(TIGER,3)
Ø68Ø END.
```

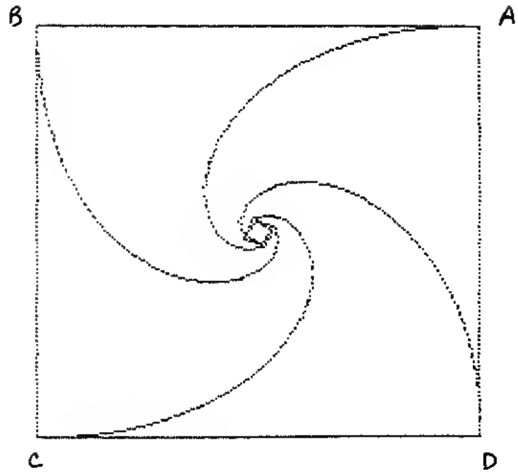


Figure 24

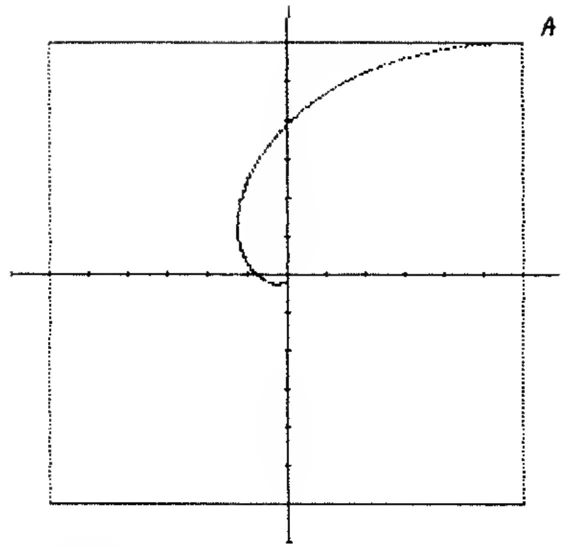


Figure 25

examples, one-quarter inch on the graph has represented one unit on the coordinate axes. Now we need one-quarter inch in the x direction to be five x units, and one-quarter inch in the y direction to be 100 y units.

In Scaler, the function to be graphed is $F(x) = x^3 + 4x^2 - 60x$. (See Fig. 18.) This function crosses the x axis at $x = -10$, 0, and 6, and has a relative minimum point at $(10/3, -118.5)$ and relative maximum point at $(-6,280)$. By graphing this function for x's in the range from -20 to 20 and for y's in the range from -400 to 400, we get a graph that shows its critical points. In this example, the relationship between the u-v coordinate system and the scaled x-y system is given by:

$u = 16(1 + H0)$ for $x = 5$, $v = 18(1 + K0)$ for $y = 100$
$u = 16(2 + H0)$ for $x = 10$, $v = 18(2 + K0)$ for $y = 200$

and so forth. Thus the relations $u = 16(x/5 + H0)$ and $v = 18(y/100 + K0)$, or $x = 5(u/16 - H0)$ and $y = 100(v/18 - K0)$ will give the desired scaling. Substituting these values for x and y into $y = f(x)$, we get $100(v/18 - K0) = f(5(u/16 - H0))$, which simplifies to $v = 18[K0 + (1/100)f(5(u/16 - H0)\ )\ ]$.

In the general case, if S is the number of x units represented by one-quarter inch and Y is the number of y units represented by one-quarter inch on the graph, then the changes of coordinates between the u-v and x-y coordinate systems are $u = 16(x/S + H0)$ and $v = 18(y/T + K0)$ and $x = S(u/16 - H0)$ and $y = T(v/18 - K0)$.

So, to graph the function $y = f(x)$ using the new scale, substituting the above equations for x and y we get $T(v/18 - K0) = f(S(u/16 - H0)\ )$, which simplifies to $v = 18[K0 + (1/T)f(S(u/16 - H0)\ )\ ]$.

As before, we use v = ROUND(v). In Scaler the function F is defined in lines 024A–0276 and the value of v is computed in line 0389. The resulting graph in Fig. 18 has marks on the x axis representing five units and on the y axis representing 100 units. In Basic, after defining the function, the equation v = INT(v + .5) accomplishes the same scaling from a Basic program.

Curves of Pursuit

Curves of pursuit are one of the most interesting and challenging types of differential equations. In this class of equations, an object A is chasing an object B—a dog chasing a cat, a hawk chasing a pigeon, or a submarine chasing a submarine. The goal of object B is to evade object A, through outrunning B or by changing its direction or its velocity.

The key to obtaining a graphic approximation of the solution is the ability to program the concept of "move object A towards object B a specified distance D." The process of sketching a graph for an equation of this type is straightforward, but often tedious. However, finding the solution of the differential equation can also be difficult. A favorite question of a teacher of mathematics posed to students is, "Is your answer reasonable?" Comparing the graph of the solution to a graph done with "finite steps" is an easy test for correctness.

In the following, I give two examples of graphic approximations of the solutions of differential equations. To tempt those readers not interested in differential equations, there is bound to be an interesting game built on the concept of moving A toward B a distance of D.

The first example I call Dogcat (Program Listing 11). In this example, suppose a cat is sitting at a point in the x-y plane. Let that point be the origin, and suppose further that to the north of the cat there is a tree, and to the east of the cat there is a dog. When the dog sees the cat, he begins to chase it; the dog always moves toward the cat a distance determined by its velocity. At the same time the cat begins to run toward the tree. If the velocities of the dog and cat are known, together with the original distances between the dog and the cat and the cat and the tree, then it is possible to determine whether or not the cat safely reaches the tree.

In Dogcat the Line procedure has been changed slightly. The interactive portion has been removed and the new version plots a line segment joining two points (X1,Y1) to (X2,Y2).

The Move procedure, called by MOVE(K,X,Y), moves the point X toward the point Y a distance of K units. The distance D between the two points X = (X1,X2) and Y = (Y1,Y2) is computed in line 52A. Recall that the distance between the two points is given by the distance formula

$$D = \sqrt{(Y1 - X1)^2 + (Y2 - X2)^2}$$

If the distance is less than or equal to K (the distance to be moved), then the point X is moved to the point Y. Otherwise, the point X is moved K units toward the point Y along the line joining these two points. The parametric equations for the line joining these two points are

$$x = (1 - M)X1 + M*Y1$$
$$y = (1 - M)X2 + M*Y2$$

In this form, M = 0 gives the point (X1,X2), M = 1 gives the point (Y1,Y2), and as M varies from 0 to 1, the point

on the line goes from the point X to the point Y. To move a distance K along this line from X, you must find the coordinates (x,y) for which the distance between this point and X is K. Using the distance formula again and solving for M gives

$$M = \frac{K}{\sqrt{(Y1 - X1)^2 + (Y2 - X2)^2}}$$

In the main body of the program (614H–814H), the initialization is done and lines 6A4H and 6C4H fix the original positions of the dog and cat. In this case, the distance between the dog and cat is 13 units and between the cat and the tree is eight units. Lines 702H and 70BH define the velocities of the dog and cat. The moving of the dog toward the cat and cat toward the tree is accomplished in the Repeat statement (lines 71BH-7CCH), which continues until the dog reaches the cat or the cat reaches the tree. The graphs resulting from two sets of velocities are shown in Figs. 19 and 20. The first shows the path of the dog in an example in which the cat reaches the tree and the second shows the path when the cat does not reach the safety of the tree.

The final example is a well-known problem. In fact, its graph was the cover of *Scientific American* a few years ago. In this problem, you have a square with its center at the origin and sides parallel to the coordinate axes. At each corner of the square there is a bug. Each bug moves counterclockwise towards the bug "in front of it;" i.e., A moves toward B, B toward C, C toward D, and D toward A. It is also assumed that the distance between each bug is 2a units, and the bugs move at the same speed.

Bugs (Program Listing 12) uses the same procedures and functions as Dogcat. A Square procedure has been added that draws the initial square. The sequence of moving each bug toward the one in front of it would theoretically take place simultaneously; however, that cannot be the case in a program. In order to model the simultaneous movement, you must remember to send bug D toward the point where bug A was at the previous instant. This is done using a point denoted OLDA, representing the previous position of A. The bug at D moves toward OLDA. Figures 21, 22, and 23 show the resulting paths for the point A for values of K equal to 2, 1, and 5.

The solution of the differential equation for the path of A is given by

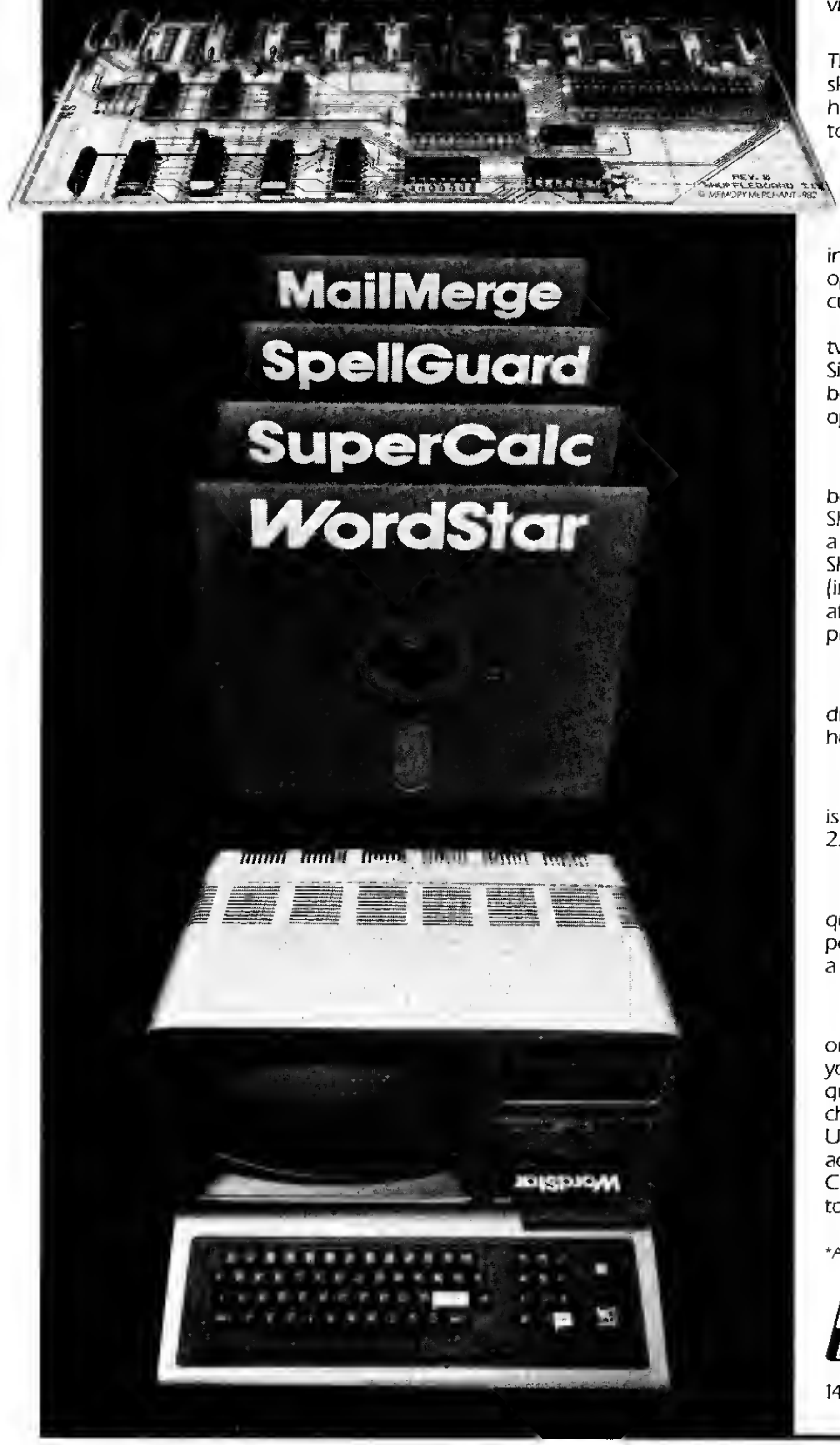
$$r = \sqrt{2a}\ e^{\left(\frac{\pi}{4} - \theta\right)}$$

in polar equations, where the original position of bug A is the point (a,a). Program Listing 13 graphs the solution to this problem and the graph is shown in Fig. 24. This program is the Pascal equivalent of POLAR/BAS. (See Part I.) Comparing the graphs of the real solution to the graphs for the various values of K, you can see the following: First, as K gets smaller, the approximations to the graph of the solution get better. Second, the value K = .5 gives a good approximation to the graph of the solution.

This article gives a set of programs that will produce the graph of many functions. However, no set of programs can be complete enough to fill every need. I hope I have provided enough insight into the programs so that they can be adapted to other situations. ■

Barbara Clinger is a professor of mathematics at Wheaton College, Norton, MA 02766.

INTRODUCING 64K CP/M 2.2 FOR MODEL III $299.



Now you can run WordStar, SuperCalc, SpellGuard, MailMerge and other CP/M based software on your TRS-80 Model III.

And walk away with incredible savings.

Memory Merchant's new Shuffleboard III is the first and only 64K CP/M 2.2 system available for your Model III.

SOPHISTICATED DESIGN

The Shuffleboard III's advanced design includes 16K of RAM, giving your Model III the power of full 64K CP/M 2.2, without interference of the ROM or the video memory.

COMPLETE CP/M SYSTEM

The Shuffleboard III is not a kit, nor a skeleton. You will receive a top quality, high performance system that is shipped to you completely assembled, burned-in, tested and insured with one of the industry's best warranties.

EASY INSTALLATION & USE

Illustrated step-by-step instructions insure fail-safe installation and reliable operation. No permanent modifications, cut traces or soldering is required.

The Shuffleboard III easily plugs into two existing sockets inside your Model III. Simple keyboard commands switch between CP/M and normal TRS-80 operation.

NO RISK TRIAL

We are so convinced that you will be absolutely delighted with your Shuffleboard III system that we offer a no risk trial. After purchasing the Shuffleboard III system, you may return it (intact) for any reason within 15 days after shipment and we will refund the purchase price (less shipping).

SHIPPED DIRECT FROM STOCK

The Shuffleboard III is shipped to you direct from stock, normally within 48 hours after receipt of your order.

INTRODUCTORY PRICE

For a limited time, Memory Merchant is offering the Shuffleboard III with CP/M 2.2 at an introductory price of only **$299.**

1 YEAR LIMITED WARRANTY

The Shuffleboard III's reliability, through quality controlled production and proven performance, has enabled us to offer a full 1 year limited warranty.

HOW TO ORDER

Please send check, money order, VISA or MASTERCHARGE (add ICA #) with your order. Sorry, no C.O.D's. Specify quantity desired. Shipping and handling charge is $5.00 per board ($9.00 outside USA & Canada)* California residents add 6% or 6½% Sales Tax where applicable. Credit card purchases may be telephoned to (415) 483-1008. Order Today!

OEM and DEALER inquiries invited.

*Air Mail shipments to Canada & all other countries.

TM Memory Merchant

14666 Doolittle Dr., San Leandro, CA 94577
Telephone (415) 483-1008 ✓461

Cobol on Your 80

by Sam Perry

Learning a new computer language can be a long tedious process, but this article leads you painlessly into the confusing world of Cobol.

Learning Cobol on the TRS-80 is much like trying to find your way through a Scott Adams adventure—everything you need is there; the problem is finding the information and then deciding what to do with it.

This article will detail the fundamentals of Cobol programming. In addition to reviewing the techniques for using Radio Shack's Cobol editor, compiler and runtime package, we will explore the basic elements of Cobol from the divisions of the Cobol program to setting up and processing sequential files.

Introduction to Cobol

Cobol (Common Business-Oriented Language) is the most widely-used business-programming language today. As much as 80 percent of all new business application programs are written in Cobol.

Although there are a number of reasons for Cobol's popularity, the two most commonly cited are the language's machine independence and self documentation.

In short, machine independence means that a Cobol program written for one computer can be easily converted to run on another. This machine independence is the result of the specifications for the language that have been laid down by the American National Standards Institute (ANSI or ANS). This means once you learn Cobol on the TRS-80, you are qualified to program in Cobol on any machine from micro to mainframe.

The self-documenting nature of the language simplifies updating and debugging, thus reducing programming costs.

Cobol's most serious drawback is its inability to handle complex mathematical computations; but then again, the language was not designed for that purpose.

Choosing a Suitable Text

Before undertaking the challenge of learning Cobol, you should acquire a suitable text. Radio Shack's Cobol editor, compiler and runtime programs make up an impressive package, but the reference manual is a weak link in the system. It is useless as an instructional text and of little value as a reference guide. Confusing and ambiguous statements such as "This variation of the Perform statement is used to augment the values referenced by one or more identifiers or index-names in an orderly fashion during the execution of a Perform statement" are common.

The book you select should be specifically for the ANSI '74 or ANS '74 version of Cobol. (ANSI meets every six years or so to decide on changes that should be made in Cobol. The version of Cobol offered by Radio Shack adheres to the 1974 standards.)

Do not let titles such as "Structured Cobol" confuse you—Cobol is Cobol. ("Structured" simply means the author will also present structured programming techniques.) I suggest any introductory book on the subject by Shelly and Cashman or Stern and Stern.

Basic Cobol Terminology

A program is merely a set of instructions given to the computer by a programmer. These instructions are coded in a computer language.

Why not give the computer instructions in English? The answer is simple. Since a computer cannot make judgments, the instructions we give it must be totally clear and singular in meaning. English is inherently ambiguous. For example, the phrase "take a seat" is clearly subject to two or more interpretations. Instructions written in a computer language have only one possible meaning.

Programs written in computer languages are referred to as "software," and the physical computer equipment is termed "hardware."

Cobol is a compiled language, while

Basic is interpreted. After keying in a Basic program you need merely type Run to execute the instructions found in the program lines. With Basic, the computer interprets and performs each instruction as it is encountered. (This is similar to a human interpreter translating line by line as a foreigner speaks.) In Cobol, the entire program is translated (or compiled) before any of it is executed. (This is similar to the interpreter translating an entire speech before delivering the message.)

The actual Cobol program lines that you type in are known as the source code. After typing in this source code it must be compiled into a form called object code. This object code is the translated version of the Cobol program. It is the version that can be executed.

Once the object code is created you can run it as often as you like without recompiling the source code.

Finally, as the source code is being compiled, the compiler lists all syntax errors in your program (lines that break the rules of the language). These errors include misspellings, illegal use of data items or variables, and improper punctuation.

While the compiler will locate and flag syntax errors, it cannot detect errors in program logic—when you tell the computer to do something other than what you intended. For example, if you tell the computer to subtract two numbers when you meant to add them, your program has a logic error.

Included in Radio Shack's Cobol package are three major programs: editor, compiler, and runtime.

To create, compile, and run a Cobol program you must understand the interrelationships between these programs, and how each is properly executed.

Following the instructions below, key in, compile and execute the Cobol program in Listing 1. You can use your right arrow on the Model III and the tab key on the Model II to tab over to the next margin as you enter the code. You should be in the TRSDOS Ready state with a copy of your Cobol disk in drive 0. When executed, this program will allow you to key in a line and specify the position on the screen where you would like to see it printed.

Step 1—The Program Editor (CEDIT)

First, you must key in the source code. To do this you need to access the program editor—simply type CEDIT and press enter while in TRSDOS. After the program editor loads you are ready to begin keying in the source code.

Although the TRS-80 Cobol reference manual outlines a variety of options available for the editor, there are only six essential commands:

I Insert
R Retype
D Delete
W Write to the disk
L Load from the disk
Q Quit the editor

To type in a Cobol program you must get into the editor's insert mode. To do this simply press the letter I. Line 000100 will appear and you are ready to begin typing the first line. While entering lines, be sure to pay strict attention to spacing because, unlike Basic, it must be precise. Refer to the margin requirements in Fig. 1 for spacing details.

After each line press enter and the next line number will appear automatically. When you complete the last line, press enter and then break to exit the insert mode.

Before writing the source code to the disk, proofread the program lines. You can view the lines by pressing the up and down arrows.

Should you find an error in a program line, the simplest way to make a correction is to retype the line. To do this simply type the letter R followed by the number of the line you wish to change. For example, when you enter R 120 the number 00120 will appear and you can retype the line.

To delete lines, simply press the letter D followed by the number of the line you wish to delete from the program. The editor automatically renumbers program lines after one is deleted on the Model III.

When you are satisfied that the source code is entered correctly, store it on the disk by typing the letter W followed by the name of your program. (Note that the name must not be longer than eight characters and should not have an extension. Also, the name must begin with a letter and consist of only letters and digits—no blanks or special characters.)

Now, to return to TRSDOS, press

Margin Requirements Chart

Columns	Use	Explanation
1-6	Sequence numbers	Used for reference to lines for purpose of insertion, change and deletion.
7	Continuation or remark	(–) means continuation of data name or literal started on a previous line. (*) means that the line serves only as a comment or remark.
8-11	Margin A	Division, Section and Paragraph names must begin in this margin.
12-72	Margin B	Most Cobol entries, particularly in the procedure division, must begin in this margin.

Coding Rules

1) Division and Section names:
   a) Must begin in Margin A
   b) Must end with a period
   c) Must appear on a line alone
2) Paragraph Names:
   a) Must begin in Margin A
   b) Must end with a period
   c) May appear on a line alone or with other entries
   d) Must always be followed by at least one space
3) Statements:
   a) Must begin in Margin B
   b) Must end with a period
   c) May appear on a line alone or with other entries
   d) Must always be followed by at least one space

Figure 1

the letter Q.

Step 2—Compiling the Program (RSCOBOL)

When the source code is safely stored on the disk, it is time to compile the program into object code. From TRSDOS, simply enter RSCOBOL filename.

Two options in RSCOBOL allow you to see the source code as it is being compiled. They are T, which displays the code on the terminal (screen), and P, which copies the code on the printer.

```
RSCOBOL PROGRAM1 T
RSCOBOL PROGRAM1 P
RSCOBOL PROGRAM1 T P
```

As the program is compiled, if any errors are found, you must return to the editor (with CEDIT) and reload the source code (with L filename). Then make any corrections with the R and D commands. Rewrite the program to the disk with the W command and then end the session with Q. You are then ready to recompile the corrected program.

Step 3—Running the Program (RUNCOBOL)

When an error-free version of the program is compiled, the object code can be executed. Do this in TRSDOS by typing RUNCOBOL filename.

After you have successfully keyed in, compiled and executed the program in Listing 1, you can check your disk directory by entering DIR. You will see

"Every Cobol program must have four divisions: Identification, Environment, Data, and Procedure."

your program listed twice—as filename/CBL and filename/COB.

When you write your source code to the disk from the editor, the extension /CBL is appended to the filename you supply. Then, when the program is compiled, the object code is written to the disk with the extension /COB. You need to use extensions with the programs only when referring to the file in a TRSDOS command such as Copy, Kill or List.

The Cobol Program

Every Cobol program must have four divisions: Identification, Environment, Data, and Procedure. These divisions are broken down into sections, which are subdivided into paragraphs, which are composed of statements. See Fig. 1 for a summary of the margin requirements and coding rules necessary for these entries.

The Identification division serves to identify the program. Program-ID is the only required entry in this division. Other entries serve merely as documentation. Note that the program ID you assign need not be the same as the name you use to write the source code to the disk, although this is generally a good practice.

In the Environment division, the computer the program is to be compiled and run on must be listed. Although the actual computer name is merely for documentation, the entries must be made. In this division you must also assign the files you plan to use to devices. For example, you might

assign an output file to the printer or a sequential file to a disk.

The Data division is where all data items (variables) used in the program must be defined. Unlike Basic, you cannot simply pick a variable out of thin air; it must be defined in this division before it can be used in the program.

Another purpose of the Data division is to allow you to describe the fields within the files you intend to use. For example, if you plan to store names and addresses on the disk, you would have to define the fields (i.e. name, street, city, state, zip code, phone number) in the record. If you plan to use no files or variables in your program, which would rarely if ever be the case, you still must list the division's name in your program.

Finally, the Procedure division is the portion of the program where the actual instructions are entered. This is where you tell the computer what it is that you want done. This division most closely resembles the Basic program.

Program Listing 2 will print the message "SHORTEST COBOL PROGRAM" on the screen. This is probably about as short a program as can be written in Cobol—all divisions are present even though there are no entries for the Data division.

Two frequently-used Cobol statements are Display and Accept. The text you select may not go into any detail on either statement, because these instructions are used primarily for displaying information on the screen and accepting it from the keyboard. Until recently, most Cobol programming was done by punching program lines into computer cards. There was no interaction between the programmer and the computer or the user and the computer. With the advent of the computer terminal and microcomputer, a truly interactive Cobol program can now be written.

Printing Messages on the Screen

A variety of items can be printed on the screen: numeric and nonnumeric literals, figurative constants, and the contents of data items.

A numeric literal is simply a value such as 10, 3, or −15. It can also be called a constant since its value never changes in the course of a program.

In contrast, a nonnumeric literal is any string of characters enclosed in quotes. Examples of nonnumeric literals are "Hello", "919 872-1955", and "a*b?/d@$". These items can include any printable character on the keyboard.

Figurative constants are Cobol words that represent the value they describe. Examples include zeros and spaces.

Finally, data names are programmer-defined terms used to represent data. In Basic they are called variables.

Since we haven't discussed the Data division and the definition of data items, this explanation of Display will be restricted to printing literals.

To print messages on the screen, refer to the RSCOBOL section of the Cobol reference manual under the Display statement. Display is the equivalent to Print in Basic. You should note the options Erase, Line, and Position are similar to Basic's CLS, and PRINT@ respectively. To print the word "HELLO" on line 10 beginning five spaces over on a clear screen you would use the following line in the Procedure division:

```
DISPLAY "HELLO" LINE 10
        POSITION 5 ERASE.
```

Look at Listings 1 and 2—all Cobol statements found in the Procedure division must be located in paragraphs.

See List of Advertisers on Page 563

A paragraph must begin with a programmer-created name that describes what the paragraph will instruct the computer to do. This is a form of self-documentation.

Note that paragraph names begin in margin A (columns 8–11) and end with a period. Statements in the paragraph must begin in margin B (columns 12–72) and also conclude with a period. As you can see from the sample listings, a statement can occupy two or more lines. It is concluded by a period.

Accepting Input from the Keyboard

Basic's powerful Input statement allows you to key in a response to a prompt. Cobol's Accept statement performs this same function.

The format for the Accept statement is as follows:

```
ACCEPT data-item.
```

When executed, the computer waits for you to type in some information. The response is then assigned to the data-item just as it is assigned to a variable in Basic.

A number of options can be used with Accept: Line, Position, Prompt, Tab, and Convert. An Accept statement asking for a person's name might look like this (note that the Display statement is necessary to print the prompt on the screen):

```
DISPLAY "ENTER YOUR NAME:"
        LINE 5 POSITION 10.
ACCEPT NAME-ITEM LINE 5 POSITION
        27 PROMPT TAB.
```

The Display statement places the prompt Enter Your Name: on line 5 of the screen starting at tab position 10. Accept then allows you to key in the name starting on line 5 at position 27 (which is at the end of the line displayed). The option Prompt causes the computer to draw a solid line where your entry will be typed. Tab tells the computer to wait until you press Enter.

You must use the Convert option when accepting numeric data that contains a decimal point. For example:

```
DISPLAY "ENTER THE AMOUNT" LINE
        12 POSITION 5.
ACCEPT AMOUNT-PAID LINE 12
        POSITION 22 PROMPT
        TAB CONVERT.
```

Before a data item (such as name-item or amount-paid) can be used in an Accept statement, it must be defined in the Data division. With your textbook, begin work on defining data items (or variables) in the Data division.

The Four Divisions

Each Cobol program consists of four divisions: Identification, Environment, Data and Procedure. While each division can contain numerous entries, we will concentrate on the most frequently used.

Figure 2 is a skeleton of the most commonly required entries. Consult this "cheat sheet" and the program listings provided as you read the following sections. (Key in capitalized items exactly as they appear, and supply all lowercase entries.)

Identification Division

Undoubtedly, the simplest division to code is the Identification division. As you can see in lines 100-150 of Fig. 2, the Identification division contains only five paragraphs. The only required entry is Program-ID. All other entries serve merely for documentation, but you should routinely include them.

The program name, which follows Program-ID, identifies the program to the computer. It should consist of no

more than eight characters and must begin with a letter. The name can be made up of only letters and digits—no spaces or hyphens are allowed. The following are valid program names: PROGRAM1, TEST and A123. Remember that this program name does not have to be the same as the name you use when writing the source code to the disk from CEDIT.

You can make all other entries in this division using any characters on the keyboard, including spaces. For example, you could enter the date as August 28, 1982 or as 8/28/82. These are the only programmer-supplied entries in which you can use blanks.

Earlier versions of Cobol allowed a Remarks entry in this division. With the ANSI '74 version, you can place remarks anywhere in the program simply by entering an asterick in column 7 of the program line, followed by a comment or remark. Remarks are used primarily for additional notes within the source code of the program. They are for the programmer's benefit and are ignored when the program is compiled.

Environment Division

Entries in the Environment division cover the computer equipment, or hardware, used to compile and execute the program. Consequently, this division is the most machine-dependent portion of the program—entries depend entirely on the computer equipment you are using. Lines 170–230 of Fig. 2 outline the division.

Although the Environment division can contain three sections, the two most commonly used are the Configuration and Input-Output sections. In the Configuration section, there are two required paragraphs: Source-Computer and Object-Computer. These paragraphs name the system on which the program is keyed in and executed, respectively. These can be, and generally are, the same.

Note that these and all other programmer-created names (other than those found in the Identification division) must conform to the rules in Fig. 3. It is important to remember that programmer-supplied names cannot contain any blanks. The hyphen is generally used in lieu of the blank.

Unlike the Configuration section, the Input-Output section is necessary only if your program will input or output information to devices such as a printer, disk drive, tape unit or card reader. Input is the movement of information into the computer by means of the keyboard, disk drives or other input device. Output, in contrast, is the flow of information from the computer to devices such as the printer or disk drives. An example of input would be reading names and addresses from a disk, while output could involve printing mailing labels.

"Output is the flow of information from the computer to devices such as the printer or disk drives."

With the TRS-80, you need only be concerned with the printer and disk drives when considering input and output.

Before any device can be used for input or output, it must be assigned a filename. This is accomplished in the File-Control paragraph with a Select clause. As is illustrated in line 230 of Fig. 2, the Cobol word Select is followed by a file name that the programmer creates; then the words ASSIGN TO are followed by the specifications of the device that the program will address. We will cover the specifications for the TRS-80 printer and disk drives later.

Data Division

The material covered in the Data division may appear difficult and perhaps abstract, but it will all come together when we get into the Procedure division. If you find this section difficult, read your textbook and then move on to the next division with confidence that the material covered here will soon become clear.

You must describe all data items (variables) and files used in the program in the Data division. As you can see in Fig. 2, this division is made up of two sections: File and Working-Storage.

A file is an overall classification of

```
00100 IDENTIFICATION DIVISION.
00110 PROGRAM-ID.          LISTING1.
00120 AUTHOR.              your name here.
00130 INSTALLATION.        your location.
00140 DATE-WRITTEN.        today's date here.
00150 SECURITY.            UNCLASSIFIED.
00160
00170 ENVIRONMENT DIVISION.
00180 CONFIGURATION SECTION.
00190 SOURCE-COMPUTER.     MODELIII-48K.
00200 OBJECT-COMPUTER.     MODELIII-48K.
00210
00220 DATA DIVISION.
00230 WORKING-STORAGE SECTION.
00240 01  STORAGE-AREAS.
00250       02 LINE-TYPED-IN          PICTURE X(50).
00260       02 POSITION-SELECTED      PICTURE 99.
00270       02 WAIT-CHARACTER         PICTURE X.
00280
00290 PROCEDURE DIVISION.
00300 PROGRAM-TITLE.
00310       DISPLAY "P R O J E C T  1"
00320       LINE 1 POSITION 23 ERASE.
00330 ACCEPT-LINE-ROUTINE.
00340       DISPLAY "TYPE IN A LINE AND PRESS <ENTER>."
00350       LINE 5 POSITION 10.
00360       ACCEPT LINE-TYPED-IN
00370       LINE 7 POSITION 10 TAB.
00380 ACCEPT-LINE-POSITION.
00390       DISPLAY "SELECT LINE NUMBER (1-15)"
00400       LINE 9 POSITION 10.
00410       ACCEPT POSITION-SELECTED
00420       LINE 9 POSITION 36 TAB.
00430 PRINT-MESSAGE-ON-LINE-SELECTED.
00440       DISPLAY LINE-TYPED-IN
00450       LINE POSITION-SELECTED POSITION 10 ERASE.
00460 ENTER-TO-CONTINUE.
00470       DISPLAY "PRESS <ENTER> TO CONTINUE OR <BREAK> TO STOP"
00480       LINE 16 POSITION 10.
00490       ACCEPT WAIT-CHARACTER
00500       LINE 16 POSITION 55.
00510 START-ROUTINE-OVER.
00520       GO TO PROGRAM-TITLE.
00530 END-PROGRAM.
00540       STOP RUN.
00550       END PROGRAM.
```

Program Listing 1. Screen printing program

data pertaining to a specific category. For example, a name and address file would contain names and addresses, similar to an address book.

A record is a single unit within a file, such as an individual's name, address, and phone number.

Fields within a record refer to the specific categories within the record. In the address book example, the fields would be name, street, city, state, zip code and phone number.

In Cobol two types of fields are used—group items and elementary items. A group item is a field that is further subdivided. For example, in our address book we could have three main fields: name, address and phone. Since address is broken down into street, city, state and zip code, it is a group item. Items which are not further subdivided, such as name, street or phone, are termed elementary items.

If the program is to address any files, either printer or disk, the files must be described in a FD (File Description) entry. Note that the letters FD must begin in margin A while the file name must be entered in margin B. Also, note that the file name appearing in the FD is the same as the name first defined in the Select clause of the Environment division. There must be one FD for each file established in the Select clauses.

Four clauses generally appear in the FD entry. (Refer to lines 280–310 of Fig. 2.) While the order in which the four clauses appear makes no difference, Block Contains integer Records is presented first. This clause specifies how records are stored and retrieved. Consult your Cobol text for details on blocking factors. On the TRS-80 a blocking factor of one should be used (i.e., Record Contains 1 Records). Despite the fact that one is singular, the word Records must be used.

The Record Contains integer Characters clause specifies the size of the data record being described. Later the record will be described; at that time its length will be determined. An example is Record Contains 120 Characters.

Although four clauses should be included in the FD entry, only one is actually required—Label Records are Standard/Omitted. This clause is used to specify whether header and trailer labels are used on tape and disk. Check your text for specific details but simply remember to always use the word Standard when describing a disk file and Omitted for a printer file. (Use Label Records are Standard for a disk file and Label Records are Omitted for a printer file.)

The final clause of the FD entry is Data Record is record-name. In this clause it is up to the programmer to give a name to the records within the file described. Remember that the file was named in a Select clause of the Environment division and now the record within that file must be given a name. An example is Data Record is Disk-Record.

Only one period is found in the FD—it follows the final clause, thus ending the sentence.

You may notice that some texts include commas between clauses in the FD and throughout the program. The comma is optional—use it anywhere to aid readability, or omit it altogether.

Following each FD entry must come a record description. Refer to lines 320–380 of Fig. 2. The record description begins with the number 01 in margin A followed by the record-name, which is coded in margin B. Actual fields within the record are defined and described in this portion of the program.

The relationship of the fields is specified by level numbers between 01 and 49. See Fig. 4 for a description of a name/address record using level numbers. In the example, note that the

```
00100 IDENTIFICATION DIVISION.
00110 PROGRAM-ID.              program-name.
00120 AUTHOR.                  programmer's name.
00130 INSTALLATION.            computer location.
00140 DATE-WRITTEN.            today's date.
00150 SECURITY.                level of security.
00160
00170 ENVIRONMENT DIVISION.
00180 CONFIGURATION SECTION.
00190 SOURCE-COMPUTER.  computer-specifications.
00200 OBJECT-COMPUTER.  computer-specifications.
00210 INPUT-OUTPUT SECTION.
00220 FILE-CONTROL.
00230        SELECT file-name ASSIGN TO device-specifications.
00240
00250 DATA DIVISION.
00260 FILE SECTION.
00270 FD     file-name
00280        BLOCK CONTAINS integer RECORDS
00290        RECORD CONTAINS integer CHARACTERS
00300        LABEL RECORDS ARE {STANDARD or OMITTED}
00310        DATA RECORD is record-name.
00320 01     record-name.
00330        02 group-item.
00340           03 elementary-item  PICTURE type(length).
00350           03 elementary-item  PICTURE type(length).
00360        02 elementary-item     PICTURE type(length).
00370        02 elementary-item     PICTURE type(length).
00380        02 FILLER              PICTURE X(length).
00390 WORKING-STORAGE SECTION.
00400 01     group-item.
00410        02 elementary-item     PICTURE type(length).
00420        02 elementary-item     PICTURE type(length) VALUE "a value".
00430        02 elementary-item     PICTURE type(length) VALUE figurative const.
00440        02 elementary-item     PICTURE type(length) value numeric literal.
00450 77     independent-data-item  PICTURE type(length).
00460
00470 PROCEDURE DIVISION.
00480 paragraph-name.
00490        COBOL statement.
00500        COBOL statement.
00510        COBOL statement.
00520 paragraph-name.
00530        COBOL statement.
00540        COBOL statement.
00550 TERMINATE-PROGRAM.
00560        STOP RUN.
00570        END PROGRAM.
```

Figure 2. Cheat Sheet

higher level numbers indicate that the fields are subsets of the items with lower level numbers. Fields with identical level numbers are independent of each other.

Remember that group items are subdivided. An example of a group item is Address. Elementary items are those fields which are not further subdivided. In the example, the elementary fields are Name, Street, City, State, Zip Code and Phone. It is up to the programmer to decide which fields are needed in a record description. Several additional examples of record descriptions will be provided in future listings.

As can be seen in Fig. 4, each elementary item has an associated Picture clause. This clause defines the length of the field and the type of data it can contain.

In the Picture clause the letter A signifies that the data in the field is alphabetic. Such a field can contain only letters and blanks.

The letter X defines an alphanumeric field, one containing any character on the keyboard including letters, digits, blanks and special characters.

Finally, 9 defines a field as numeric. Such a field can contain only digits, a sign and a decimal point. This is the only type data that can be used in mathematical operations.

The length of the data field can be specified in one of two ways. To signify an alphanumeric field three characters in length, the Picture could be XXX or X(3). To maintain consistency, the second format is generally recommended.

Looking at Fig. 4, the Name field is alphanumeric and 20 characters long. The Street and City fields are also alphanumeric and 20 characters long. A two-character alphanumeric field is set aside for State, while a five-character alphanumeric field is defined for Zip Code.

Many people question the use of alphanumeric for Zip Code. They argue that the zip code is a number and should be defined as numeric. While Zip Code can be correctly defined as numeric, you should generally define a field as numeric only if the program performs mathematical calculations using the value. There is no need to add, subtract, multiply or divide a zip code, therefore there is no reason for defining it as numeric. The same logic applies to the Phone field, which is also defined as alphanumeric.

You can determine the overall record length by totaling the length of each field in Fig. 4. This total (80) appears in the Record Contains integer Characters clause of the Data division's FD entry. In this case the clause would be Record Contains 80 Characters.

In line 380 of Fig. 2, the Cobol word Filler is found. This reserved word (a word that has a predefined meaning)

> *"You should define a field as numeric only if the program performs mathematical calculations using the values."*

can be used to define any field that will not be addressed by name in the Procedure division. It can be used only in the Data division, and must be defined as alphanumeric. The sample listings here and later will clarify the use of Filler.

In the Working-Storage section you must define all data items (variables) to be used for purposes other than input or output to files. Lines 390–450 of Fig. 2 provide a skeleton of what this section might look like. It is difficult to know which items you will need in this section until you begin preparing the Procedure division.

In general, here is where counters and accumulators are initialized, page headings are assigned to data items and general-purpose data items are named and described.

This section differs from the file section in that an initial value can be assigned to each data item. This value is an alphabetic or alphanumeric value enclosed in quotes and called a nonnumeric literal. (Note that some texts show the nonnumeric literal enclosed in single quotes. On the TRS-80 it is necessary to always use double quotes.) The value can also be a numeric value, which is called a numeric literal, or a figurative constant, such as zeros or spaces.

In the Working-Storage section, group items and elementary items are established using the level numbers 01–49. Level 77 can be used in this section to define an independent item—one having no relationship to any group item. Generally all data items in this section can be included as subordinate to a group item even if the group item has a name such as General-Purpose-Items or Work-Areas.

Procedure Division

In this final division of the Cobol program, you give the computer a set of instructions to carry out. These instructions are included in paragraphs, subdivided into sentences and statements.

Each instruction or set of related instructions must begin with a paragraph

1) Name can be from 1 to 30 characters in length.
2) Allowable characters include:
   a) letters
   b) numbers
   c) hyphens
3) No embedded blanks are allowed or other punctuation.
4) At least one character must be a letter.
5) Name cannot begin or end with a hyphen.
6) No two hyphens can be used together.
7) Names must be unique.
8) Names cannot be reserved words.
9) Names should be meaningful.

Figure 3. Rules for forming programmer-supplied names

```
00300 01  DISK-RECORD.
00310     02  NAME              PICTURE X(20).
00320     02  ADDRESS.
00330         03  STREET        PICTURE X(20).
00340         03  CITY          PICTURE X(20).
00350         03  STATE         PICTURE X(2).
00360         03  ZIP-CODE      PICTURE X(5).
00370     02  PHONE             PICTURE X(13).
```

Figure 4. Lengths of each field

name conforming to the rules for programmer-supplied names shown in Fig. 3. A paragraph name should describe the purpose or function of the statements enclosed in the paragraph. The more descriptive, the better. Valid paragraph names include Display-Heading-on-Screen, or Calculate-Payroll-Deductions.

In theory, a paragraph should perform only one function. If this is the case, anytime a particular function needs to be performed, control can be sent to the paragraph needed.

The Procedure division is much like a Basic program. As a matter of fact, many Cobol statements are very similar to those found in Basic. Figure 5 provides a chart of similar Cobol and Basic statements.

There is no set of rules for coding the Procedure division. No two programs will have exactly the same Procedure division, although they may have similar routines.

One routine which must appear in each Procedure division is the paragraph to end the program. In Fig. 2, this paragraph is named Terminate-Program, although you could name it anything you like. The Cobol statement Stop Run makes program execution cease. The optional instruction End Program tells the compiler that it has reached the end of the source-code listing.

To further illustrate the statements and techniques required in the Procedure division (and throughout the Cobol program), let's examine a sample program that asks the user to enter his name and address and then sends this information to the printer.

Program Listing 3 contains the source code of the Cobol program. Before output can be sent to the printer, a number of entries must be made throughout the program. Here we will trace the entries necessary to create a printer file, describe a print record and output to the printer.

The first reference to the print file occurs in a Select clause of the Environment division as is shown in line 230. For the TRS-80, the line Select file name Assign to Print, "Printer" must be used. It is up to you to supply the file name. (Programmers often use the name PRNT as their printer file name.)

Another reference to the printer file must be made in an FD entry of the File section of the Data division, as shown in lines 270–310. As you recall, any programmer-supplied name can be used for the data record name. In line 310 we chose the name Print-Line since each record will represent one line on the printer.

Following the FD comes the record description. When setting up a printer file, this entry can be quite simple. As shown in line 320, the record can be simply a line 132 alphanumeric characters long. It is not necessary to define any fields in the Print-Line—data can simply be moved into this Print-Line and output to the printer.

On some printers it is necessary to define the record as containing 133 characters, with the first representing a carriage return. This should not be done on the TRS-80, though—define your Print-Line as having 132 characters.

The next reference to the printer file is made in the Procedure division. Here you must follow four steps to enable the program to output to the printer.

First, you must open the file. The Open statement allows the program to gain access to the file. (It is similar to opening a file cabinet drawer before having access to the files inside.)

The format for the Open statement is shown in line 410. Note that the word Open is followed by Output. This sets the printer file up as an output file. The final word in the Open statement is the name of the file you wish to open—PRNT, in this case.

After opening the file you must move the information you want printed to the Print-Line. This is done through the Move statement. In line 560 the name you typed, which has been stored in a data item, is moved to the Print-Line.

Once the information to be printed has been moved to the Print-Line, it can be output to the printer with the Write statement. In line 570 the Write statement sends the contents of the Print-Line to the printer.

The final statement required is Close. This statement instructs the computer to make the file no longer available for access. Close the file only after you have output all that is necessary. The format for closing a printer file is shown in line 630—Close PRNT.

As I pointed out earlier, the Open statement is used to gain access to the file named. It can refer to either a printer or disk file.

The Move statement is used to copy the contents of one data item (the sending field) into another (the receiving field). The contents of the sending field are unchanged after the move, while the original contents of the receiving field are destroyed and replaced by those from the sending field. We will discuss additional details on the Move statement in the section on editing data items.

After data to be sent to the output file has been moved to an output record such as the Print-Line, it can be sent out to the file with the Write statement.

The Write statement instructs the computer to output the contents of the record specified to the file with which it is associated. In Listing 3, the record Print-Line is associated with the printer file named PRNT. When Write Print-Line is executed, the contents of the record Print-Line are output to the PRNT file.

Finally, the Close statement is used to terminate access to a particular file. All files must be closed before the program is ended.

Line by Line

Line 100—this division name is always the first line in the program.

Line 110—Program-ID is the only required entry in the Identification division. The name following must be eight characters or less and contain only letters and numbers.

Lines 120–150—these lines provide details for documentation purposes. These entries can contain any character on the keyboard, including blanks.

Line 160—here we left a blank line between divisions to aid readability of the program.

Line 170—the second division name appears here.

Line 180—this is always the first section in the Environment division.

Lines 190–200—these two paragraphs name the equipment on which the program will be written and executed. Usually both entries are identical.

Line 210—the second section in this

Program Listing 2. Shortest Cobol program

```
00100 IDENTIFICATION DIVISION.
00110 PROGRAM-ID.     LISTING2.
00120
00130 ENVIRONMENT DIVISION.
00140 CONFIGURATION SECTION.
00150 SOURCE-COMPUTER.    MODELII-64K.
00160 OBJECT-COMPUTER.    MODELII-64K.
00170
```

Listing 2 continues

Listing 2 continued

```
00180 DATA DIVISION.
00190
00200 PROCEDURE DIVISION.
00210 DISPLAY-PARAGRAGH.
00220      DISPLAY "SHORTEST COBOL PROGRAM" ERASE.
00230 TERMINATE-PROGRAM.
00240      STOP RUN.
00250      END PROGRAM.
```

```
00100 IDENTIFICATION DIVISION.
00110 PROGRAM-ID.                    PROJECT2.
00120 AUTHOR.                        student's name.
00130 INSTALLATION.                  JTC.
00140 DATE-WRITTEN.                  04/21/82.
00150 SECURITY.                      UNCLASSIFIED.
00160
00170 ENVIRONMENT DIVISION.
00180 CONFIGURATION SECTION.
00190 SOURCE-COMPUTER.               MODELII-64K.
00200 OBJECT-COMPUTER.               MODELII-64K.
00210 INPUT-OUTPUT SECTION.
00220 FILE-CONTROL.
00230      SELECT PRNT ASSIGN TO PRINT, "PRINTER".
00240
00250 DATA DIVISION.
00260 FILE SECTION.
00270 FD   PRNT
00280      BLOCK CONTAINS 1 RECORDS
00290      RECORD CONTAINS 132 CHARACTERS
00300      LABEL RECORDS ARE OMITTED
00310      DATA RECORD IS PRINT-LINE.
00320 01   PRINT-LINE                       PICTURE X(132).
00330 WORKING-STORAGE SECTION.
00340 01   ADDRESS-IN-FIELDS.
00350      03 NAME-IN                       PICTURE X(30).
00360      03 STREET-IN                     PICTURE X(30).
00370      03 CITY-STATE-IN                 PICTURE X(30).
00380
00390 PROCEDURE DIVISION.
00400 OPEN-FILE-TO-PRINTER.
00410      OPEN OUTPUT PRNT.
00420 DISPLAY-AND-ACCEPT-DATA.
00430      DISPLAY "ENTER NAME:"
00440      LINE 3 POSITION 10 ERASE.
00450      ACCEPT NAME-IN
00460      LINE 3 POSITION 22 PROMPT "." TAB.
00470      DISPLAY "ENTER STREET:"
00480      LINE 5 POSITION 8.
00490      ACCEPT STREET-IN
00500      LINE 5 POSITION 22 PROMPT "." TAB.
00510      DISPLAY "ENTER CITY & STATE:"
00520      LINE 7 POSITION 2.
00530      ACCEPT CITY-STATE-IN
00540      LINE 7 POSITION 22 PROMPT "." TAB.
00550 MOVE-AND-PRINT-DATA.
00560      MOVE NAME-IN TO PRINT-LINE.
00570      WRITE PRINT-LINE.
00580      MOVE STREET-IN TO PRINT-LINE.
00590      WRITE PRINT-LINE.
00600      MOVE CITY-STATE-IN TO PRINT-LINE.
00610      WRITE PRINT-LINE.
00620 CLOSE-PRINTER-FILE.
00630      CLOSE PRNT.
00640 TERMINATE-PROGRAM.
00650      STOP RUN.
00660      END PROGRAM.
```

Program Listing 3. A name printing program

Program Listing 4. Storing names in a sequential file

```
00100 IDENTIFICATION DIVISION.
00110 PROGRAM-ID.                    SEQOUT.
00120 AUTHOR.                        SAM PERRY.
00130 INSTALLATION.                  JTC.
00140 DATE-WRITTEN.                  APRIL 4, 1982.
00150 SECURITY.                      UNCLASSIFIED.
00160
00170 ENVIRONMENT DIVISION.
00180 CONFIGURATION SECTION.
00190 SOURCE-COMPUTER.               MODELIII-48K.
```

Listing 4 continues

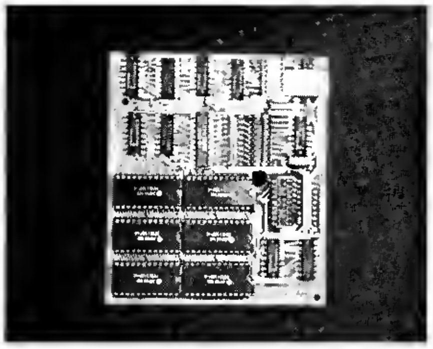
184

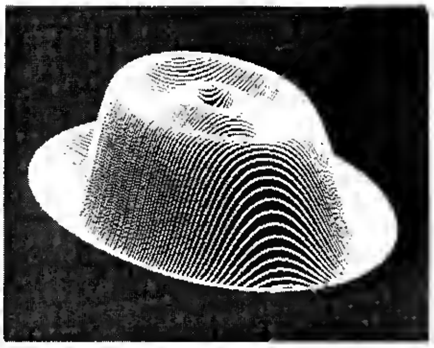
Listing 4 continued

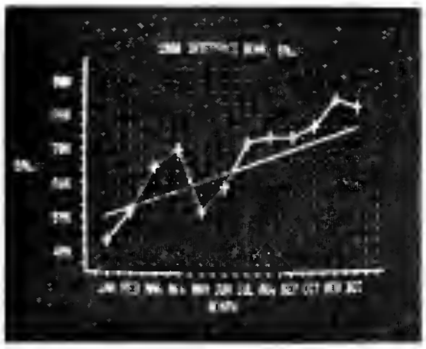
```
OBJECT-COMPUTER.         MODELIII-48K.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT NAME-FILE
    ASSIGN TO OUTPUT "NAMEFILE/SEQ".

DATA DIVISION.
FILE SECTION.
FD  NAME-FILE
    BLOCK CONTAINS 1 RECORDS
    RECORD CONTAINS 30 CHARACTERS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS NAME-RECORD.
01  NAME-RECORD.
    02 NAME-FIELD        PICTURE X(30).

PROCEDURE DIVISION.
OPEN-SEQUENTIAL-DISK-FILE.
    OPEN OUTPUT NAME-FILE.
DISPLAY-PROMPT.
    DISPLAY "ENTER 'END' AS LAST NAME"
    LINE 2 POSITION 20 ERASE.
    DISPLAY "ENTER NAME"
    LINE 6 POSITION 14.
ACCEPT-NAME.
    ACCEPT NAME-FIELD
    LINE 6 POSITION 28 PROMPT "." TAB.
    IF NAME-FIELD = "END"
      GO TO CLOSE-DISK-FILE.
WRITE-TO-DISK.
    WRITE NAME-RECORD.
LOOP-ROUTINE.
    GO TO ACCEPT-NAME.
CLOSE-DISK-FILE.
    CLOSE NAME-FILE.
TERMINATE-PROGRAM.
    STOP RUN.
    END PROGRAM.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.              SEQIN.
AUTHOR.                  SAM PERRY.
INSTALLATION.            JTC.
DATE-WRITTEN.            APRIL 5, 1982.
SECURITY.                UNCLASSIFIED.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.         MODELIII-48K.
OBJECT-COMPUTER.         MODELIII-48K.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT NAME-FILE
    ASSIGN TO INPUT "NAMEFILE/SEQ".

DATA DIVISION.
FILE SECTION.
FD  NAME-FILE
    BLOCK CONTAINS 1 RECORDS
    RECORD CONTAINS 30 CHARACTERS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS NAME-RECORD.
01  NAME-RECORD.
    02 NAME-FIELD        PICTURE X(30).

PROCEDURE DIVISION.
OPEN-SEQUENTIAL-DISK-FILE.
    OPEN INPUT NAME-FILE.
DISPLAY-HEADING.
    DISPLAY "READING IN NAMES"
    LINE 2 POSITION 24 ERASE.
READ-FROM-DISK.
    READ NAME-FILE
      AT END GO TO CLOSE-DISK-FILE.
PRINT-NAME-ON-SCREEN.
    DISPLAY NAME-RECORD.
LOOP-ROUTINE.
    GO TO READ-FROM-DISK.
CLOSE-DISK-FILE.
    CLOSE NAME-FILE.
TERMINATE-PROGRAM.
    STOP RUN.
    END PROGRAM.
```

Program Listing 5. Record Retrieval







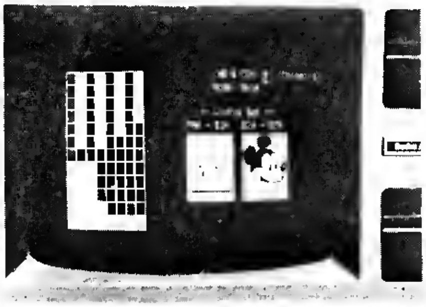
Mod III **GRAFYX SOLUTION**™ $299.95 ✓464

- 512 × 192 Dot Graphics increases resolution 16× (better than Apple)
- Extensive Business, Personal, Educational, and Scientific applications
- 14 BASIC commands set and reset Points, Lines, Circles, Boxes, etc.
- Allows Display and Printout of detailed Line, Bar, and Pie graphs
- 80 Character/line display compatible with BASIC programs and DOS
- Price includes 98K bits memory, 30 programs, 52 page manual
- Plug-in, clip-on board eliminates soldering for easy installation

REBATE **$40** REBATE

VALID 11-1 TO **12-31-82**

After purchasing Grafyx Soln or 80-Grafix, return this coupon with registration form to receive rebate



Mod I, III **80-GRAFIX**™ $169.95 ✓465

- Create and Save 128 programmable hi-resolution characters/shapes
- Characters are generated within an 8 × 12 matrix (Mod I: 6 × 12)
- Great for creating game figures, symbols, and alternate character sets
- Excellent software support (20 programs) & extensive documentation
- Module easily installs within the computer with no soldering

Manual only $15
Products guaranteed
Dealers welcome

MICRO-LABS, INC. 214-235-0915
902 Pinecrest, Richardson, Texas 75080

FREE shipping
Prepaid or COD
Tx. res add 5%

division, required only if disk or printer files are to be used.

Line 220—this paragraph name is always required after the entry in line 210.

Line 230—here the Select clause allows you to create a file name and assign it to a device (printer or disk drive). In this case, a printer file is established.

Line 240—blank to aid readability.

Line 250—the third division name appears here.

Line 260—if any files are named in Select clauses they must be described here in the File section.

Line 270—FD (File Description) entry begins with FD followed by the file name created in the Select clause.

Line 280—this line defines the blocking factor. On the TRS-80, always use one.

Line 290—specifies the number of characters in the record.

Line 300—Omitted must always be used with printer files while Standard must be used with disk files.

Line 310—in this line the record in the file is named. This is a programmer-supplied name.

Line 320—following the FD must come the record description, which begins with 01 followed by the record name just defined. While the record is often subdivided into fields, it is not necessary to do so with a printer record. This record is simply defined as having 132 characters, which is the length of the line on most printers.

Line 330—here the name of the second section in the division appears.

Line 340—this programmer-supplied group item name describes the data items which follow.

Line 350—this data item will hold the name that the user types in. Remember that data items named here in the Working-Storage section are not part of input or output. Although this item is used to allow the user to key in a name and is later moved to the Print-Line for output, it is not actually part of the input or output record. Print-Line is the only record used in this program.

Lines 360–370—these two fields will hold the user's street, city and state address.

Line 380—blank to aid readability.

Line 390—the fourth division name appears here.

Line 400—this is a paragraph name.

Line 410—the Cobol statement Open is used to gain access to the printer file.

Line 420—this paragraph name describes the function of the following statements.

Line 430–440—these two lines make up a Display statement. While the entire statement could have been placed on one line, it is more readable like this—the period in line 440 marks the end of the statement. The Display instruction tells the computer to place on the screen the item following the word Display. The line clause specifies the line number on the screen where the message should be placed. The Position clause tells the computer how far to tab over before displaying the message. Finally, Erase clears the screen before printing the message.

Lines 450–460—these two lines make up an Accept statement. Again, the period in line 460 marks the end of the statement. The Accept statement tells the computer to pause while the user types in a response (in this case, to the message printed by lines 430–440). This response is then placed in the data item specified after the word Accept (Name-In, in this case). Line tells the computer the screen line number where the keyed-in response is to be displayed. Position specifies how far to tab before displaying the response. Prompt "." tells the computer to prompt the user by placing a string of periods on the screen where the response is to be typed. Any character can be used inside the quotes. If no quotes or characters are used, Prompt displays a solid line.

Lines 470–480—see lines 430–440.

Lines 490–500—see lines 450–460.

Lines 510–520—see lines 430–440.

Lines 530–540—see lines 450–460.

Line 550—this paragraph name describes the function of the statements included in it.

Line 560—the Move instruction copies the contents of the first item (sending field) into the second item (the receiving field). In this case the contents of Name-In are moved into the Print-Line. Remember that Name-In contains the name the user enters. This Move step is necessary before the record Print-Line can be written to the printer.

Line 570—Write moves the contents of the record specified out to the file. In this case the contents of Print-Line are moved out to the printer and thus printed.

Line 580—once again, Move is used to move the contents of the first data item to the second. Here the contents of Street-In are moved to the Print-Line. This destroys the previous contents of Print-Line and replaces them with the new item.

Line 590—again, the contents of the Print-Line are sent to the printer.

Lines 600–610—this is the same as lines 560–570 and lines 580–590, except this time the contents of the data item City-State-In are moved to the Print-Line and sent out to the printer.

Line 620—here the paragraph name specifies what the instructions following will do.

Line 630—the Close statement closes the file specified. This must be done before the program is terminated. If we wanted to print other information we would not have closed the file yet.

Line 640—here the final paragraph name specifies the actions to be taken.

Line 650—Stop Run causes the program to stop running. The execution is complete.

Line 660—this line tells the compiler that this is the end of the program.

Now is the time to write your first Cobol program. Using the material presented so far, and relying heavily on Fig. 2, code a simple program. If your system includes a printer, create a printer file and output to it. Perhaps write a simple program that asks the user a series of questions and prints the responses on the printer. If you get error messages, consult your Cobol manual.

Probably the most common use of the computer in the business environment is for storage and retrieval of information. The computer has a distinct advantage over the file cabinet—it can be programmed to store and recall information, and also process the records and present them in a report format.

Sequential Files

A disk file is much like a folder kept in a file cabinet—it contains records pertaining to some particular category. For example, a sales file may contain information on the performance of a company's sales force.

File, record, and field are the three key terms used when referring to disk files.

A sequential file is one of three common Cobol types. It is perhaps the simplest and, therefore, generally the first learned.

The term "sequential" actually refers to the access mode used for storing and retrieving records from the disk. Records are stored and recalled in the order in which they are entered.

As you can see, there are disadvantages in this method of disk access. For example, if you want to retrieve the 900th record, you have to read the preceding 899. This can be time-consuming, even for a computer.

Probably the best way to visualize the notion of sequential access is to think of a music album stored on tape. When listening to the album on a tape player,

to hear the last song, you must play through all the songs preceding it. This is unlike a record album, where you can lift the needle and place it on the last track. This second type of access is termed direct, relative, or random.

If you wish to hear the entire album, there is no disadvantage in sequential access. In the same sense, if you plan to process all of the records in your disk file, there is no disadvantage in sequential access. Sequential files are only a disadvantage when you must retrieve an individual record—then they can be inefficient.

Creating a Sequential File

A number of entries are required in a Cobol program to create a sequential disk file. These entries are similar to those used to set up a printer file because a printer file is actually a sequential file.

Program Listing 4 stores names on the disk in a sequential file. The program is written in the traditional approach, as opposed to the structured approach, which we will discuss later.

The first reference to the sequential disk file is found in a Select clause of the Environment division's File-Control section, as is shown in lines 230–240.

Following the Cobol word Select comes a file name created by the programmer. After the Assign To portion of the clause must come the word Output, Input or Input-Output. The word Output tells the computer that the file will be used in this particular program only for output—records will be written to the file but not retrieved from it. Input means that this program will only input, or read, records from the file. Input-Output signifies that the file will be used for both input and output in this program.

The final entry in the Assign To clause is the actual name that the file will be stored under on the disk. This does not have to be the same as the file name listed after the word Select, which is the name used inside the program to refer to the file.

In the example we selected the disk file name NAMEFILE/SEQ. This is part of the device specifications and must adhere to TRSDOS filespecs requirements. In brief, a TRSDOS file name can consist of eight characters or less followed by a slash and up to three more characters. The name must begin with a letter, but can also contain numbers. No blanks or other characters are allowed in the name. We chose NAMEFILE/SEQ because NAMEFILE refers to the contents of the file, while SEQ reminds us that the file is sequential. In this case the full device-specifications entry is Output "NAMEFILE/SEQ".

The next reference to the disk file comes in the Data division in a FD entry. Lines 280–320 present a typical disk-file FD entry. Note the entry Label Records Are Standard—remember that Standard is required with disk files while Omitted is used with printer files. We chose the name Name-Record in this case to represent a disk record, since the file will be made up of names.

Lines 330–340 make up the record description, which here is simply a record containing one field. The Picture clause shows that Name-Field will contain 30 alphanumeric characters.

Finally, the disk file is referenced in the Procedure division. There it is treated like a printer file—we use the statements Open, Move, Write and Close to gain access to the file and output records to it.

Just as in a printer file, the disk file must be opened before it can be accessed. It can be opened as an Output file, Input file or I-O (Input-Output) file. Do not forget that the Select clause specified how the file could be used.

With Open Output Name-File, the disk file is opened for output only. When the file is opened as an Output file, you can use it only to receive records output from the computer.

The first record output from the disk file in the Output mode will be placed in the first position in the file. If the file did not exist on the disk before the Open Output statement, then it is created. If the file did already exist on the disk, it is opened and its previous contents are lost. An existing file, whose contents should not be destroyed, must be opened with either Input or I-O. We'll discuss these modes later.

Two statements you can use for placing a value into a data item are Move and Accept. When we set up a printer file (see Listing 3) data was accepted and then moved to the printer record. Now we'll show how the move can sometimes be avoided and information accepted directly into a field of the record.

In lines 450–460 we used the data item Name-Field in an Accept statement. When the user types in the name, it is placed immediately in the record field. Line 500 shows that the record can then be written directly to the disk.

Often it is necessary to define data items in the Working-Storage section that will be used to accept input from the keyboard. The information accepted by these data items can then be moved into the appropriate fields. In this example we have simply taken a short cut.

When no other records are to be added to the file, it must be closed as is shown in line 540.

In the Procedure division of Listing 4 you'll find two new statements: If and Go To.

Before exploring these Cobol instructions, let's look at the overall purpose of the program in Listing 4 and the steps required to achieve this goal.

Figure 6 is a flowchart of the program. A flowchart is a graphic representation of the steps in a computer program. The flowchart is made up of symbols, lines and arrows. The symbols represent the type of operation while the lines and arrows show the flow of execution.

An oval symbol represents the beginning or end of a program. The rectangle signifies a process such as Open, Close or Move, while the trapezoid means an input or output operation such as Read, Write, Display or Accept. Finally, a diamond represents a decision to be made with statements such as If ... Else and Go To ... Depending.

The Program

First, the disk file is opened. Then, a prompt to the user to enter a name is displayed on the screen.

Next, through the Accept statement, the user is allowed to type in the name he wishes stored on the disk.

The decision block shows that if the name End is entered the program is to be terminated; if it is not End, the name entered is to be stored on the disk.

After the record is written to the disk, control of the program returns to the paragraph in which the user is asked to enter a name. Here the user can enter another name or the word End.

You should be able to find the paragraphs in Listing 4 corresponding to each symbol on the flowchart.

The Cobol statements If and Go To appear in lines 470-480. Line 470, which says, If Name-Field = "END", simply means that if the value accepted for Name-Field is the word End, then do whatever comes next (before the period). If Name-Field does equal End then the instruction on line 480 is to be executed. If it does not equal End then the remainder of the sentence (up to the period) is to be ignored and the following sentence executed. The instruction in line 480 is indented two spaces merely to aid readability—this indicates that it is to be executed only when

the If condition is true.

In summary, the Cobol word If is always followed by a condition. If that condition is true then the remainder of the If sentence is executed. When the condition is not true, the remainder of the sentence is ignored and the program continues with the next sentence. Any number of instructions can follow the condition in the If statement, not just one as is the case in this program.

When the condition in line 470 is true, the instruction in 480 is executed. In line 480 the instruction Go To Close-Disk-File tells the computer to jump to the paragraph named Close-Disk-File. The statement Go To means just what it says—go to the paragraph named and do not do anything else on the way. Control is transferred directly to the paragraph named. The instructions found there are executed and the program continues from that point.

In Listing 4 we used the Go To statement again in the Loop-Routine to transfer control back to the Accept-Name paragraph.

You should generally avoid the Go To statement whenever possible because it completely transfers control to another part of the program. The unrestricted use of Go To can lead to a program that is difficult to debug or even read. See the section on structured programming for more details.

Sequential Input

Now that we've covered the techniques for setting up a sequential output disk file and storing information on the disk, let's focus on the details needed to input, or to read the records back from, the disk file.

Program Listing 5 reads the records created by the program in Listing 1 and displays them on the screen.

To treat the file as an input file, you must change only one entry before the Procedure division. Line 240 in Listing 5 tells the computer that the file will be used for input only in this program.

Before the file can be accessed, it must be opened as shown in line 380. Note that the file is opened as an Input file.

After the file is opened, this program displays a message on the screen to let the user know that records are to be read from the disk.

Just as the Write statement is used to output to a file, the Read statement retrieves a record from the disk (see line 430). Since the access mode is sequential, the records are read in the order they were stored—first in, first out (FIFO).

It is important to note that the disk file name is used with the Read instruction while the record name is used with Write. That is to say, you write a record but read a file.

Also, note that the Read statement contains an At End clause (see line 440). (Remember that a sentence does not end until the period. Lines 430–440 make up one instruction.)

Always use the At End clause with the Read statement to tell the computer what is to be done after the last record in the file has been read. This clause is similar to the If statement in that it is only executed when the condition is true.

The Read statement's At End clause is activated when the computer tries to read a record after all have been read. The instructions following the At End are then, and only then, executed.

As can be seen in lines 430–440, when the computer tries to read beyond the last record in the file, control is transferred to the Close-Disk-File paragraph and the program is terminated.

After the record is read, line 460 displays the contents of the Name-Field on the screen. Note that the Display statement can be used without the Line and Position clauses—without them, the item is simply displayed on the next available line on the screen.

Line 480 sends control of the program back to the Read statement, where another record is retrieved from the disk file. This loop continues until the At End condition is met.

Structured Programming Techniques

The programs in Listing 4 and Listing 5 were written in the traditional approach—execution flows from top to bottom with little concern for structure.

In an effort to bring about a more standardized method of programming, structured programming was developed.

Structured programming is a technique in which a program is written in modules. Each module performs a single function. Whenever a particular function needs to be executed, the paragraph containing that module is executed. Control then returns to a main-logic paragraph which directs the program sequencing (see Listing 6).

Line 480 executes the Display-Prompt paragraph, and then control returns to line 490.

In 490 a more sophisticated and powerful Perform instruction is employed—Perform ... Until. Here the statement instructs the computer to execute the Read-From-Disk-And-Display paragraph until the EOF-Flag equals one.

When the Perform ... Until statement is used, the paragraph named is executed and then the condition is tested. If it is not true, the paragraph is again executed and tested. This continues until the condition is true, at which time control moves on to the sentence after the Perform instruction.

In the case of line 490, the paragraph Read-From-Disk-And-Display will continue to be executed until the EOF-Flag equals 1. Remember that it was initially set to zero in line 420.

The Read-From-Disk-And-Display paragraph first displays the contents of the Name-Field last read from the disk. In line 570 it then attempts to read the next record. If a valid record is retrieved from the disk, line 490 causes the paragraph to be executed again. If the end of file is reached and the At End condition is true, the value 1 is moved into the data item EOF-Flag. Then, when line 490 tests the value of the EOF-Flag it finds it equal to one and moves on to line 500.

In the Read-From-Disk-And-Display paragraph, it is essential that the Display statement precede the Read statement. This is because the condition in line 490 is tested only after the entire paragraph has been executed. Therefore it is important that the final instruction in the paragraph be to place the one in the EOF-Flag data item—this prevents displaying the last record twice. That is why the first Read must be executed in the Mail-Logic paragraph.

The purposes behind structured programming are many. Its primary goal is to make it easier for one programmer to read or modify another's work. Also, debugging a program is easier since individual modules can be tested apart from the entire program. Finally, a team of programmers can work on a single program, with each writing individual modules that will later be merged.

An example of a structured program is in Listing 6. This program performs exactly the same task as the one in Listing 5, but the program is controlled by a single paragraph—Main-Logic.

A comparison of the two listings reveals several changes. Lines 110–150 of Listing 6 contain remarks, signified by the asterisk in column 7 of each line.

The next change is lines 400–420 of Listing 6—a Working–Storage section is present. In this section, a data item named EOF–Flag is created for use in the Procedure division. Note that the item is given an initial value of zero.

The Procedure division of Listing 6 is controlled by a Main–Logic paragraph that directs the flow of the pro-

gram. A line-by-line analysis of this paragraph will clarify techniques of structured programming, as well as introduce a new Cobol statement.

Line 460 simply opens the disk file for input.

In line 470 the first record from the disk file is read. Note the At End clause is present in case the file contains no records. Although this is highly unlikely, a good program should be prepared to handle any plausible situation.

The Perform statement is introduced in line 480. This statement is similar to the Go To instruction—it sends control of the program to the paragraph named. However, after the named paragraph is executed, control returns to the instruction following the Perform statement. This instruction allows the Main-Logic paragraph to execute modules throughout the program and still maintain control.

Printing the Report

With the few Cobol statements we have covered, it is possible to store data on the disk, retrieve it, and output it to the printer in a report format. While we'll cover additional details for printing reports later, let's consider a few fundamentals. All references apply to Listing 7.

First, we need a program that creates both disk and printer files. To do this we will merely combine the entries covered in the sections on printer and disk files.

Note the two Select clauses in lines 290–300 as well as the two FDs in lines 340–460. These are just like those presented in the sections on disk and printer files.

The technique of defining page headings in the Working-Storage section is introduced in lines 480–540. This technique allows you to create page and column headings here and print them later simply by moving the item into the Print-Line and then writing it to the printer.

A closer look at Page-Heading, created in lines 480–510, shows how the heading is to be printed. Line 490 defines the first 10 columns of the heading as spaces. Following these 10 spaces comes the actual title defined in line 500. This title is 21 characters long and has the value "N A M E R E P O R T." Finally, 101 spaces follow the title, using the entire 132 characters of the Print-Line.

It is always a good habit to define headings as 132 characters long. When the heading is moved into the Print-Line, it will clear out any other characters that may have been stored in the line.

ACCEPT	INPUT
ADD	+
AND	AND
CLOSE	CLOSE
COMPUTE	LET
DISPLAY	PRINT
DIVIDE	/
GO TO	GOTO
GO TO / DEPENDING	ON / GOTO
IF / ELSE	IF / THEN / ELSE
MOVE	LET or LSET & RSET
MULTIPLY	*
OCCURS	DIM
OPEN	OPEN
OR	OR
PERFORM	GOSUB
PERFORM / VARYING / FROM / BY / UNTIL	FOR / NEXT / STEP
READ	GET
STOP RUN	END
SUBTRACT	—
WRITE	PUT

Figure 5. Partial listing of Cobol vs. Basic statements

```
00100 IDENTIFICATION DIVISION.
00110*****************************************
00120*  * * *   S E Q U I N 1   * * *        *
00130*     READ RECORDS FROM THE DISK        *
00140*     Using Structured Approach         *
00150*****************************************
00160 PROGRAM-ID.                SEQIN1.
00170 AUTHOR.                    SAM PERRY.
00180 INSTALLATION.              JTC.
00190 DATE-WRITTEN.              APRIL 5, 1982.
00200 SECURITY.                  UNCLASSIFIED.
00210
00220 ENVIRONMENT DIVISION.
00230 CONFIGURATION SECTION.
00240 SOURCE-COMPUTER.           MODELIII-48K.
00250 OBJECT-COMPUTER.           MODELIII-48K.
00260 INPUT-OUTPUT SECTION.
00270 FILE-CONTROL.
00280     SELECT NAME-FILE
00290     ASSIGN TO INPUT "NAMEFILE/SEQ".
00300
00310 DATA DIVISION.
00320 FILE SECTION.
00330 FD  NAME-FILE
00340     BLOCK CONTAINS 1 RECORDS
00350     RECORD CONTAINS 30 CHARACTERS
00360     LABEL RECORDS ARE STANDARD
00370     DATA RECORD IS NAME-RECORD.
00380 01  NAME-RECORD.
00390     02 NAME-FIELD        PICTURE X(30).
00400 WORKING-STORAGE SECTION.
00410 01  WORK-AREAS.
00420     02 EOF-FLAG          PICTURE 9(1)  VALUE 0.
00430
00440 PROCEDURE DIVISION.
00450 MAIN-LOGIC.
00460     OPEN INPUT NAME-FILE.
00470     READ NAME-FILE AT END MOVE 1 TO EOF-FLAG.
00480     PERFORM DISPLAY-PROMPT.
00490     PERFORM READ-FROM-DISK-AND-DISPLAY UNTIL EOF-FLAG = 1.
00500     CLOSE NAME-FILE.
00510     STOP RUN.
00520 DISPLAY-PROMPT.
00530     DISPLAY "READING IN NAMES"
00540     LINE 2 POSITION 24 ERASE.
00550 READ-FROM-DISK-AND-DISPLAY.
00560     DISPLAY NAME-RECORD.
00570     READ NAME-FILE AT END MOVE 1 TO EOF-FLAG.
```

Program Listing 6. An example of a structured program

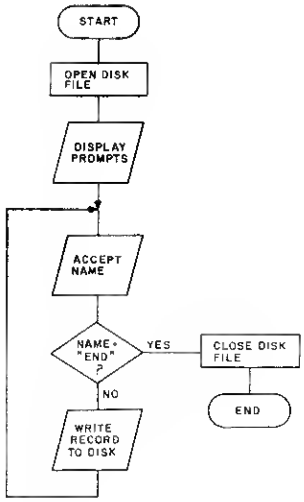


Figure 6. Flowchart of Program Listing 4

Remember, when programming on computers other than the TRS-80, you may have to define the Print-Line as having 133 characters. When this is the case, always define the first character as a space to allow for the carriage-return character.

In the description of the headings, note the use of the Cobol word Filler in place of field names. Filler is to be used in the Data division to define any field that will not be addressed by name in the Procedure division. The headings will only be referred to by their 01 group name—field names are unnecessary. See lines 610–640 for clarification.

The Procedure division begins by opening the disk and printer files. Although both files can be opened with a single Open instruction, it is more readable to use two instructions on separate lines. Note that the disk file is opened as an Input file, while the printer file is opened for Output.

Headings are output to the printer in lines 600–640. Line 640 includes a variation on the Write statement. Write to the printer can be used with an After Advancing or Before Advancing (integer) Lines clause. This clause is used to advance the printer the specified number of lines either before or after the contents of the Print-Line are printed.

In line 720, the After Advancing 2 Lines clause is used to double-space between names.

According to the ANSI '74 standards the integer in the After Advancing clause can be any number from 1 to 100. On the TRS-80 you can use values higher than 100.

Finally, note the Move instruction used in line 690. This statement moves spaces into the Print-Line to clear it before use. Since the Name-Field accounts for only the left 30 columns of the Print-Line, there is a chance that the right 102 columns could contain characters from a previous Move.

Some Homework

Write a program to store and retrieve names and addresses on a disk. The information should then be printed in mailing-label format.

Numeric Data

Up to this point, we've said little about numeric data. All our Move and Accept statements have used alphanumeric data items to manipulate nonnumeric literals. Here, we'll consider the special rules that apply to numeric items.

First, let's examine the procedure for defining numeric data items in the Data division. As you may recall, numeric data items are accompanied by a Picture clause consisting of 9s. A Picture of 9(3) represents a data item that can contain a value consisting of three digits, such as 312 or 999. Such a field can also contain values less than 100, such as 23 or 76 but these values are actually stored as 023 and 076.

To define a numeric data item that contains decimal places, use the character V where the decimal point should appear. For example, Picture 999V99 represents a numeric item with three places on the left of the decimal and two on the right. You could also write this clause as Picture 9(3)V9(2).

When using numeric data items containing a decimal, remember that the decimal is not actually stored as part of the value. For example, the value 321.28 would be stored in memory and on the disk as 32128. Therefore, the Picture clause 9(3)V9(2) would have a field length of five. The decimal point is only implied, and is not actually stored in the data item.

A numeric-data item containing a decimal place can be assigned a value in the Working-Storage section, using the following format: Picture 9(3) V9(2) Value 123.45. Be sure to include the decimal point in the Value clause.

This method of storing a decimal value without the decimal point was designed to conserve memory and disk space. This convention requires additional programming effort, however.

Program Listing 7

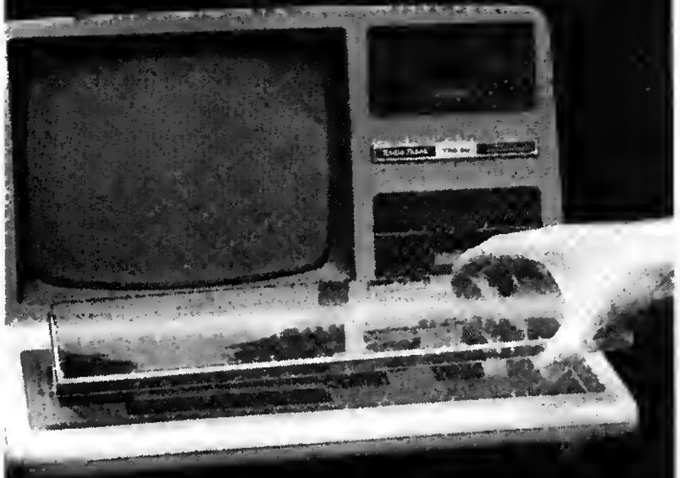
```
00100 IDENTIFICATION DIVISION.
00110*****************************************
00120*  * * *   R E P O R T 1   * * *        *
00130*     READ RECORDS FROM THE DISK        *
00140*     AND PRINT A SIMPLE REPORT         *
00150*     Using Traditional Approach        *
00160*****************************************
00170 PROGRAM-ID.                REPORT1.
00180 AUTHOR.                    SAM PERRY.
00190 INSTALLATION.              JTC.
00200 DATE-WRITTEN.              APRIL 5, 1982.
00210 SECURITY.                  UNCLASSIFIED.
00220
00230 ENVIRONMENT DIVISION.
00240 CONFIGURATION SECTION.
00250 SOURCE-COMPUTER.           MODELIII-48K.
00260 OBJECT-COMPUTER.           MODELIII-48K.
00270 INPUT-OUTPUT SECTION.
00280 FILE-CONTROL.
00290     SELECT NAME-FILE ASSIGN TO INPUT "NAMEFILE/SEQ".
00300     SELECT PRNT ASSIGN TO PRINT, "PRINTER".
00310
00320 DATA DIVISION.
00330 FILE SECTION.
00340 FD  NAME-FILE
00350     BLOCK CONTAINS 1 RECORDS
00360     RECORD CONTAINS 30 CHARACTERS
00370     LABEL RECORDS ARE STANDARD
00380     DATA RECORD IS NAME-RECORD.
00390 01  NAME-RECORD.
00400     02 NAME-FIELD  PICTURE X(30).
00410 FD  PRNT
00420     BLOCK CONTAINS 1 RECORDS
00430     RECORD CONTAINS 132 CHARACTERS
00440     LABEL RECORDS ARE STANDARD
00450     DATA RECORD IS PRINT-LINE.
00460 01  PRINT-LINE             PICTURE X(132).
00470 WORKING-STORAGE SECTION.
00480 01 PAGE-HEADER.
00490     02 FILLER        PICTURE X(10) VALUE SPACES.
00500     02 FILLER        PICTURE X(21) VALUE "N A M E   R E P O R T".
00510     02 FILLER        PICTURE X(101) VALUE SPACES.
00520 01 SUB-HEADER.
00530     02 FILLER        PICTURE X(6)  VALUE "NAMES:".
```

Listing 7 continues

Keep it Clean.

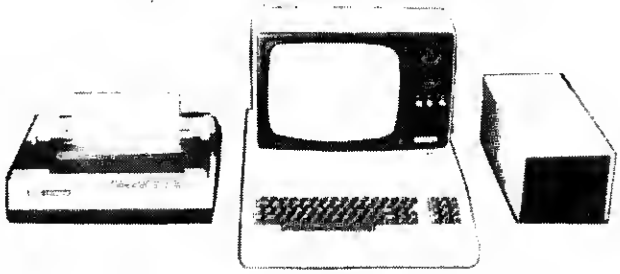
Protect Your Investment

Protect your TRS-80 Model III keyboard with our dustcover made from rigid smoked bronze plastic. Its low silhouette was custom designed to fit with the TRS-80 Model III's contemporary design, and provides the protection your keyboard requires.



Send check or money order for $12.00 to:
Nat Hellman III, Inc., 5951 East Firestone Blvd.,
South Gate, CA 9[illegible] ✓55

LNW80 COMPUTERS



IN STOCK NOW!! CALL FOR SPECIAL PRICES ON LNW80s, NEC PRINTERS, RGB COLOR MONITORS, AND TANDON DISK SYSTEMS.

***** NEW LNW SOFTWARE! *****

EXCELLONIX

7180 WOODROW WILSON DR
LOS ANGELES, CA 90068

(213) 650-5754 - (714) 973-1939

CopyArt II

WILL CHANGE THE WAY YOU THINK ABOUT YOUR TRS-80!

THE MOST POWERFUL WORD PROCESSOR AND ALL PURPOSE COMPUTER PROGRAM AVAILABLE FOR THE TRS-80.

LOOK AT ALL THESE FEATURES

1. INSERT characters, words, lines, paragraphs or other files.
2. DELETE characters, words, lines, paragraphs.
3. COLUMNS. CopyArt II can be instructed to print your text from one to six columns. Super easy to use! No complicated commands. Great for doing newsletters, magazine layouts etc. NO MORE CUT AND PASTE!
4. SORTING. Sort lines of text by any field. Sorts up to 650 items in less then 7 seconds. Sort indices, table of contents, names, words or whatever in descending or ascending order. Used with CopyArt's math function it is great for small Inventories, Receivables, Payables etc.
5. Screen widths from 32-255 characters wide. Screen widths can be changed to allow formatting your text as you want.
6. MATH. Built in MATH function for doing calculations on columns or rows. Used with the SORT command, CopyArt II can do a small inventory of 200-300 items, or keep track of small receivables or payables, general ledgers or home financial reports. Super floating point precision up to 32 digits!
7. *GRAPHICS. CopyArt has a built in graphics program that allows inserting graphics within your text. Drawings, graphs, illustrations, cartoons etc. may be used within newsletters or company reports. Graphics commands include: Plot between points, Circles, Squares, Fill, Erase, Draw, Move, Pixel cursor controls and more.
8. *GRAPHIC CHARACTERS. CopyArt has a built in graphics character generator. Used for typesetting large letters from 3 to 25 times normal size! Yes, you can even print characters down the page as well as across. Black on white or white on black.
9. JUSTIFICATION is fully supported. *Proportional spaced justify is supported.
10. *SUPER or SUB-SCRIPT.
11. UNDERLINING.
12. BOLDFACING.
13. *CHANGE CHARACTER SIZE or PITCH within your document. Character size changes for dot matrix printers with capability. Pitch change for daisy wheel printers with capability.
14. HELP. Help is available for all the commands at the touch of a key while using the word processor. Super for training inexperienced secretaries. Great reminder for experienced people as well. MENU DRIVEN Help for over 45 commands.

CopyArt II DOES ALL THIS... & MUCH, MUCH MORE!

COD and Credit Cards
CALL TOLL FREE to order:
1-800-528-1149

SIMUTEK

Computer Products Inc. ✓163

SIMUTEK COMPUTER PRODUCTS INC., 4897 E. SPEEDWAY BLVD., TUCSON, AZ 85712, (602) 323-9391
DEALER, DISTRIBUTOR, & PRINTER/MANUFACTURER INQUIRIES INVITED
TRS-80 and Scripsit are TM of Radio Shack a Tandy corp.
*Indicates printer must have capability to do function.

✓See List of Advertisers on Page 563

For example, when you print or display a numeric-data item, the decimal point will not be included. You must edit all numeric data before output to the screen or printer—we'll discuss editing later.

If a numeric-data item contains a negative value, then the Picture clause describing the item must be signed—it must contain the character S. Picture S9(3) describes a numeric-data item that can contain a positive or negative three-digit number.

When calculating the length of a signed field, you should count the S character. In the example above, the field length is four. (When punching computer cards, an additional column is not needed to record the sign, but on the TRS-80 the space is required for the plus or minus sign.)

Accepting numeric data from the keyboard presents a special set of problems.

First, when accepting numeric data from the keyboard, a Convert clause must be used in the Accept statement. The Convert clause is used to align the decimal point typed in by the user with the V in the numeric data item's Picture. A sample Accept statement is as follows: Accept Number-1 Line 10 Position 10 Prompt Convert Tab. (Remember that the order of the clauses in the Accept statement makes no difference.)

Forgetting to use the Convert clause when accepting numeric data can cause unpredictable results.

(While Convert is not a standard ANSI Cobol word, it is also used on systems other than the TRS-80.)

An Accept statement using a data item with a Picture of 9(3)V9(2) will allow you to enter only five characters, including the decimal point. Therefore, even though the data item is designed to hold three digits to the left of the decimal place and two to the right, a value this large cannot be keyed in. Perhaps the simplest solution to this problem is to define a Picture larger than that actually needed.

The best way to understand all of the details involved in accepting numeric data is to write a sample program which tests various combinations of Picture clauses.

Editing Numeric Data

Before numeric data can be output to the screen or printer, it must be edited. Editing allows you to define exactly how the values will be displayed—you can suppress leading zeros, insert dollar signs, and add commas. A numeric value that is stored on the disk as 123456789 can be printed as $1,234,567.89—clearly a far more meaningful value.

Numeric values are edited by moving them from a numeric-data item into an edit-data item. All edit-data items are defined in the Working-Storage Section.

Examples of editing operations are shown in Program Listing 8. Note that the numeric data item is moved into the edit data item and then the edit data item is printed or displayed.

Figure 7 contains a summary of the edit symbols, along with the results of some sample editing.

The edit character Z suppresses leading zeros. A value of 000123 can be printed as 123 with this character in the edit Picture.

Generally, it is not a good idea to fill the entire edit Picture with Zs—a value of zero would be displayed as a blank. Include at least one 9 in the Picture clause (use Picture ZZZ.99 rather than Picture ZZZ.ZZ).

The decimal point is used in the edit Picture to tell the computer where to align the implied decimal point of the numeric-data item with the actual decimal point of the edit-data item. For example, a numeric-data item with a Picture of 999V9 and a value of 1234 can be moved into an edit-data item with a Picture of ZZZZ.99. The edited result is the value 123.40—the decimal points align themselves.

The dollar-sign edit character is used to place a $ before a number. If you use more than one $, they act as zero-suppression characters. For example, when the value 1234 is moved into the edit field $$$$$.99, the result is $12.34. If the same value is moved into the field $ZZZZ.99, the result is $ 12.34.

The comma-edit character is used to insert a comma in the appropriate location of a value. For example, the value 123456 can be moved into an edit-data item with a Picture 999,999. When the edit item is sent to the screen or printer, the value 123,456 is displayed.

Asterisks can be used in the edit Picture to replace leading zeros to be replaced by asterisks. This is done primarily when printing paychecks.

A leading or trailing plus or minus sign can be added to a value by including − or + in the edit Picture. If a minus sign is included, it is printed only when the value is negative. If the value is positive, no sign is printed. When the plus sign is used in the edit Picture, + is printed with the value when it is positive or unsigned and − when it is negative.

The symbols CR (credit) and DB (debit) can be printed with a negative value simply by ending the edit Picture with these letters. Note that these symbols are printed only with negative values.

Although there are additional edit symbols, the final one we'll consider is B for blank. Placing a B in the edit field in-

Symbol	Meaning
Z	Substitute blanks for leading zeroes
.	Insert decimal point
,	Insert comma
+	Insert + sign if value is positive and − sign if value is negative
−	Insert − sign if value is negative
$	Insert $ sign as first character
CR	Append CR to value if item is negative
DB	Append DB to value if item is negative
*	Insert asterisk in place of leading zero
B	Insert a blank in this position

Sending Field		Receiving Field	
Picture	Contents	Edit Picture	Edited Results
9(3)V9(2)	12345	$ 999.99	$ 123.45
9(3)V9(2)	00123	$ ZZZ.99	$ 1.23
9(3)V9(2)	00123	$ $$$$.99	$1.23
9(5)V9(2)	0000123	$**,***.99	$****1.23
S9(6)	−123456	+ZZZ,ZZZ	−123,456
9(9)	246871977	999B99B9999	246 87 1977
S9(3)	−123	ZZZ−	123−
S99V99	−1234	$ ZZ.99CR	$ 12.34CR
99V99	0000	$ ZZ.99	$ .00

Figure 7. A summary of Edit symbols

Listing 7 continued

```
00540      02 FILLER    PICTURE X(126) VALUE SPACES.
00550
00560 PROCEDURE DIVISION.
00570 OPEN-DISK-PRINTER-FILE.
00580      OPEN INPUT NAME-FILE.
00590      OPEN OUTPUT PRNT.
00600 PRINT-HEADINGS.
00610      MOVE PAGE-HEADER TO PRINT-LINE.
00620      WRITE PRINT-LINE.
00630      MOVE SUB-HEADER TO PRINT-LINE
00640      WRITE PRINT-LINE AFTER ADVANCING 2 LINES.
00650 READ-RECORD-FROM-FILE.
00660      READ NAME-FILE
00670      AT END GO TO TERMINATE-PROGRAM.
00680 MOVE-DATA-TO-PRINT-LINE.
00690      MOVE SPACES TO PRINT-LINE.
00700      MOVE NAME-FIELD TO PRINT-LINE.
00710 WRITE-REPORT-ITEM.
00720      WRITE PRINT-LINE AFTER ADVANCING 2 LINES.
00730 LOOP-ROUTINE.
00740      GO TO READ-RECORD-FROM-FILE.
00750 TERMINATE-PROGRAM.
00760      STOP RUN.
00770      END PROGRAM.
```

Add

Format 1	Add item-1 To item-2.			
	or Add item-1 item 2 To item-3.			
Example	Add AMT-1 AMT-2 To AMT-3.			
Before		2	4	6
After		2	4	12
Format 2	Add item-1 To item-2 Giving item-3.			
	or Add item-1 item 2 To item-2 Giving item-3.			
Example	Add AMT-1 To AMT-2 Giving AMT-3.			
Before		2	8	6
After		2	8	10

Subtract

Format 1	Subtract item-1 From item-2.			
	or Subtract item-1 item-2 From item-3.			
Example	Subtract AMT-1 AMT-2 From AMT-3.			
Before		6	3	20
After		6	3	11
Format 2	Subtract item-1 From item-2 Giving item-3.			
Example	Subtract 3 From 7 Giving Result-1.			
Before		3	7	12
After		3	7	4

Multiply

Format 1	Multiply item-1 By item-2.			
Example	Multiply Amount-1 By Amount-2.			
Before		3	5	
After		3	15	
Format 2	Multiply item-1 By item-2 Giving item-3.			
Example	Multiply AMT-1 By AMT-2 Giving AMT-3.			
Before		6	5	10
After		6	5	30

Divide

Format 1	Divide item-1 Into item-2.			
Example	Divide AMT-1 Into AMT-2.			
Before		2	6	
After		2	3	
Format 2	Divide item-1 {By/Into} item-2 Giving item-3.			
Example	Divide AMT-1 Into AMT-2 Giving AMT-3.			
Before		2	6	18
After		2	6	3

Figure 8. General formats for the Add instruction

✓See List of Advertisers on Page 563

serts a blank in the location it occupies. For example, a social-security number stored as the value 123456789 could be moved into an edit data item with the Picture 999B99B9999. When the edit item is output to the printer or screen it would look like this: 123 45 6789.

Arithmetic Operations

The arithmetic operations available in Cobol include Add, Subtract, Multiply, Divide, and Compute.

The Add instruction is used to perform addition on numeric-data items. There are two general formats for the Add instruction (see Fig. 8).

The first format allows you to sum data items. All data items in the instruction are first summed, and then the result is moved into the data item following the word To. The only item in the operation whose value is changed is the final data item. This is the case with all operations involving Add, Subtract, Multiply and Divide.

In the second format the clause Giving is included in the instruction. When Giving is used in any arithmetic operation, the data item following the word receives the result of the operation. The example in Fig. 8 under the second format of the Add statement shows that the items to the left of the word Giving are summed and the result placed in the final data item.

Since the data item following Giving does not actually participate in the operation, and only acts as a receiving field, it can be an edit-data item. All other items must be either numeric-data items or numeric literals. This is the case in all arithmetic statements.

Subtract is used to perform subtraction on numeric items. As with the Add statement, there are two formats for Subtract. Figure 8 shows that the first format simply sums all of the data items and numeric literals before the word From and then subtracts this amount from the data item following From.

In format 2, the Giving clause is added. In this case, the subtraction is performed and the result moved into the data item following the word Giving. As with the Add statement and all other arithmetic statements, the data item following Giving does not participate in the operation; it acts only as a receiving field.

This is a good place to point out two general rules about the Cobol arithmetic statements Add, Subtract, Multiply and Divide.

First, only the value of the final data item is ever changed. If this final data item follows the word Giving, it does not participate in the actual arithmetic operation but acts only as a receiving field. When the Giving clause is not included, the final data item participates in the operation and then acts as the receiving field for the result.

Second, a data item following the word Giving can be a numeric or edit-data item, while all others must be either numeric-data items or numeric literals.

The Multiply statement is limited because only two values can be multiplied in a single instruction. Nonetheless, there are two formats for the statement (see Fig. 8).

In the first format, the two values are multiplied and the result is placed in the final data item. The second format is used to multiply the two items to the left of the word Giving and then place the result in the final data item.

With Divide, as with the other arithmetic statements, there are two general formats. The first format shown in Fig. 8 divides the value represented by item-1 into the value represented by item-2. The result is then stored in item-2.

The second format causes the two data items to the left of the word Giving to be divided; the result is placed in the final data item. Note that when using this second format, item-1 can be divided by or into item-2. For example, if 2 is divided into 6 the quotient is 3. If 2 is divided by 6 the quotient is .333.

The final, and most powerful, of the Cobol arithmetic statements is the Compute instruction. Figure 9 is a list of the arithmetic symbols used with Compute as well as some examples using the statement.

With the Compute statement, the data item to the left of the equals sign always acts as the receiving field. It does not take part in the computation. For example, Compute Sum = 2 + 2, would make Sum equal to 4. The computation to the right of the equal sign is completed and the result is moved into the data item to the left of the sign.

When using the Compute instruction to perform complex calculations, remember the mathematical hierarchy of operations as is shown in Fig. 9. For example, Compute Average = 10 + 20 + 30 / 3, does not calculate the average. As can be seen in the hierarchy of operations, multiplication and division are performed before addition and subtraction. In this example the data item Average would equal 40. To actually calculate the average, you must use parentheses to instruct the computer to perform the addition before the division: Compute (10 + 20 + 30) / 3.

Two additional options can be used with all arithmetic statements. They are Rounded and On Size Error.

You can use the Rounded option to round a number off to a specified decimal place. For example, the statement Subtract Discount From Amount

Program Listing 8. Examples of editing operations

```
000100 IDENTIFICATION DIVISION.
000110**************************************
000120*  COBOL 1  -- BUDGET PROJECT          *
000130*        Module 1  -- Store Data       *
000140*       Using Traditional Approach     *
000150**************************************
000160 PROGRAM-ID.                  WORK6.
000170 AUTHOR.                      student's name.
000180 INSTALLATION.                JTC.
000190 DATE-WRITTEN.                MAY 1982
000200 SECURITY.                    UNCLASSIFIED.
000210
000220 ENVIRONMENT DIVISION.
000230 CONFIGURATION SECTION.
000240 SOURCE-COMPUTER.             MODELIII-48K.
000250 OBJECT-COMPUTER.             MODELIII-48K.
000260 INPUT-OUTPUT SECTION.
000270 FILE-CONTROL.
000280     SELECT BUDGET-FILE ASSIGN TO OUTPUT "BUDGET/SEQ".
000290
000300 DATA DIVISION.
000310 FILE SECTION.
000320 FD  BUDGET-FILE
000330     BLOCK CONTAINS 1 RECORDS
000340     RECORD CONTAINS 44 CHARACTERS
000350     LABEL RECORDS ARE STANDARD
000360     DATA RECORD IS BUDGET-RECORD.
000370 01  BUDGET-RECORD.
000380     02 EMPLOYEE-NUMBER       PICTURE X(5).
000390     02 EMPLOYEE-NAME         PICTURE X(25).
000400     02 BUDGET-AMOUNT         PICTURE 9(5)V99.
000410     02 CURRENT-EXPENSE       PICTURE 9(5)V99.
000420 WORKING-STORAGE SECTION.
000430 77  RESPONSE                 PICTURE X.
000440
000450 PROCEDURE DIVISION.
000460 OPEN-OUTPUT-DISK-FILE.
000470     OPEN OUTPUT BUDGET-FILE.
```

Listing 8 continues

Listing 8 continued

```
000480 DISPLAY-PROMPTS.
000490      DISPLAY "I N P U T   B U D G E T   D A T A"
000500      LINE 1 POSITION 15 ERASE.
000510      DISPLAY "EMPLOYEE NUMBER:"
000520      LINE 5 POSITION 8.
000530      DISPLAY "EMPLOYEE NAME:"
000540      LINE 7 POSITION 10.
000550      DISPLAY "AMOUNT BUDGETED:"
000560      LINE 9 POSITION 8.
000570      DISPLAY "CURRENT EXPENSE:"
000580      LINE 11 POSITION 8.
000590 ACCEPT-BUDGET-DATA.
000600      ACCEPT EMPLOYEE-NUMBER
000610      LINE 5 POSITION 25 PROMPT "." TAB.
000620      ACCEPT EMPLOYEE-NAME
000630      LINE 7 POSITION 25 PROMPT "." TAB.
000640      ACCEPT BUDGET-AMOUNT
000650      LINE 9 POSITION 25 PROMPT "." CONVERT TAB.
000660      ACCEPT CURRENT-EXPENSE
000670      LINE 11 POSITION 25 PROMPT "." CONVERT TAB.
000680 WRITE-DATA-TO-DISK.
000690      WRITE BUDGET-RECORD.
000700 ASK-IF-MORE-DATA.
000710      DISPLAY "MORE DATA TO ENTER (Y/N) ?"
000720      LINE 15 POSITION 10.
000730      ACCEPT RESPONSE
000740      LINE 15 POSITION 38 TAB.
000750      IF RESPONSE EQUAL "Y"
000760      GO TO DISPLAY-PROMPTS.
000770 TERMINATE PROGRAM.
000780      CLOSE BUDGET-FILE.
000790      STOP RUN.
000800      END PROGRAM.
```

Compute

Symbols Used With Compute

+	=	Add
−	=	Subtract
*	=	Multiply
/	=	Divide
**	=	Exponentiate (Not supported on TRS-80)

Hierarchy of Operations

1.	( )	Items in parentheses are evaluated first
2.	**	Exponentiations are performed next
3.	*/	Multiplication and division occur next
4.	+ −	Finally, addition and subtraction

Examples using Compute

Compute Amount-1 = 35.

This statement moves the value 35 into the data item Amount-1.

Compute Tax = .05 * Amount.

Amount is multiplied by .05 with the product being moved into the data item Tax.

Compute Total = AMT-1 + AMT-2 + AMT-3.

Here AMT-1, AMT-2 and AMT-3 are totaled. The sum is moved into Total.

Compute Average = (AMT-1 + AMT-2 + AMT-3) / 3.

The three amounts are added first and then the sum is divided by three. This amount is then moved into Average.

Figure 9. Arithmetic symbols used with Compute

Program Listing 9.

```
000100 IDENTIFICATION DIVISION.
000110****************************************
000120*  COBOL I  -- BUDGET PROJECT          *
000130* Module II -- Print Report From Data  *
000140*      Traditional Method              *
000150****************************************
000160 PROGRAM-ID.              WORK5.
000170 AUTHOR.                  student's name.
000180 INSTALLATION.            JTC.
```

Listing 9 continues

Listing 9 continued

```
000190 DATE-WRITTEN.            MAY 1982
000200 SECURITY.                UNCLASSIFIED.
000210
000220 ENVIRONMENT DIVISION.
000230 CONFIGURATION SECTION.
000240 SOURCE-COMPUTER.         MODEL III-48K.
000250 OBJECT-COMPUTER.         MODEL III-48K.
000260 INPUT-OUTPUT SECTION.
000270 FILE-CONTROL.
000280     SELECT BUDGET-FILE ASSIGN TO INPUT "BUDGET/SEQ".
000290     SELECT PRNT ASSIGN TO PRINT "PRINTER".
000300
000310 DATA DIVISION.
000320 FILE SECTION.
000330 FD  BUDGET-FILE
000340     BLOCK CONTAINS 1 RECORDS
000350     RECORD CONTAINS 44 CHARACTERS
000360     LABEL RECORDS ARE STANDARD
000370     DATA RECORD IS BUDGET-RECORD.
000380 01  BUDGET-RECORD.
000390     02 EMPLOYEE-NUMBER       PICTURE X(5).
000400     02 EMPLOYEE-NAME         PICTURE X(25).
000410     02 BUDGET-AMOUNT         PICTURE 9(5)V99.
000420     02 CURRENT-EXPENSE       PICTURE 9(5)V99.
000430 FD  PRNT
000440     BLOCK CONTAINS 1 RECORDS
000450     RECORD CONTAINS 132 CHARACTERS
000460     LABEL RECORDS ARE OMITTED
000470     DATA RECORD IS PRINT-LINE.
000480 01  PRINT-LINE      PICTURE X(132).
000490 WORKING-STORAGE SECTION.
000500 01 HEADERS.
000510     02 HEADER-1.
000520         03 FILLER            PICTURE X(35) VALUE SPACES.
000530         03 LITERAL1          PICTURE X(13) VALUE "BUDGET REPORT".
000540         03 FILLER            PICTURE X(84) VALUE SPACES.
000550     02 HEADER-2.
000560         03 FILLER            PICTURE X(5) VALUE SPACES.
000570         03 LITERAL2          PICTURE X(10) VALUE "EMP #".
000580         03 LITERAL3          PICTURE X(30) VALUE "EMPLOYEE NAME".
000590         03 LITERAL4          PICTURE X(12) VALUE "BUDGET AMT".
000600         03 LITERAL5          PICTURE X(16) VALUE "CURRENT EXP".
000610         03 LITERAL-6         PICTURE X(59) VALUE "BALANCE".
000620     02 DETAIL-LINE-1.
000630         03 FILLER                 PICTURE X(5) VALUE SPACES.
000640         03 EMPLOYEE-NUMBER-OUT    PICTURE X(5).
000650         03 FILLER                 PICTURE X(5) VALUE SPACES.
000660         03 EMPLOYEE-NAME-OUT      PICTURE X(25).
000670         03 FILLER                 PICTURE X(5) VALUE SPACES.
000680         03 BUDGET-AMOUNT-OUT      PICTURE Z(4)9.99
000690         03 FILLER                 PICTURE X(5) VALUE SPACES.
000700         03 CURRENT-EXPENSE-OUT    PICTURE Z(4)9.99
000710         03 FILLER                 PICTURE X(5) VALUE SPACES.
000720         03 BALANCE-OUT            PICTURE Z(6)9.99CR.
000730         03 FILLER                 PICTURE X(49) VALUE SPACES.
000740     02 DETAIL-LINE-2.
000750         03 FILLER                 PICTURE X(64).
000760         03 LITERAL-7              PICTURE X(7) VALUE "TOTAL".
000770         03 TOTAL-OUT              PICTURE Z(6)9.99CR.
000780         03 FILLER                 PICTURE X(49) VALUE SPACES.
000790 01  WORK-AREAS.
000800     02 TOTAL-COUNTER         PICTURE S9(7)V99 VALUE ZERO.
000810     02 BALANCE               PICTURE S9(7)V99 VALUE ZERO.
000820
000830 PROCEDURE DIVISION.
000840 OPEN-DISK-PRINT-FILE.
000850     OPEN INPUT BUDGET-FILE.
000860     OPEN OUTPUT PRNT.
000870 PRINT-HEADINGS.
000880     MOVE HEADER-1 TO PRINT-LINE.
000890     WRITE PRINT-LINE BEFORE ADVANCING 3 LINES.
000900     MOVE HEADER-2 TO PRINT-LINE.
000910     WRITE PRINT-LINE BEFORE ADVANCING 2 LINES.
000920 READ-RECORD-FROM-FILE.
000930     READ BUDGET-FILE
000940     AT END GO TO MOVE-TOTAL-TO-PRINT-LINE.
000950 MOVE-DATA-TO-PRINT-LINE.
000960     MOVE EMPLOYEE-NUMBER TO EMPLOYEE-NUMBER-OUT.
000970     MOVE EMPLOYEE-NAME TO EMPLOYEE-NAME-OUT.
000980     MOVE BUDGET-AMOUNT TO BUDGET-AMOUNT-OUT.
000990     MOVE CURRENT-EXPENSE TO CURRENT-EXPENSE-OUT.
001000 CALCULATE-BALANCE.
001010     SUBTRACT CURRENT-EXPENSE FROM BUDGET-AMOUNT GIVING BALANCE.
001020 MOVE-BALANCE-TO-PRINT-LINE.
001030     MOVE BALANCE TO BALANCE-OUT.
001040 ADD-BALANCE-TO-TOTAL.
001050     ADD BALANCE TO TOTAL-COUNTER.
001060 WRITE-REPORT-ITEM.
001070     WRITE PRINT-LINE FROM DETAIL-LINE-1.
001080 LOOP-ROUTINE.
001090     GO TO READ-RECORD-FROM-FILE.
001100 MOVE-TOTAL-TO-PRINT-LINE.
001110     MOVE TOTAL-COUNTER TO TOTAL-OUT.
001120 PRINT-FINAL-TOTAL.
001130     WRITE PRINT-LINE FROM DETAIL-LINE-2
001140     AFTER ADVANCING 2 LINES.
001150 TERMINATE-PROGRAM.
001160     CLOSE BUDGET-FILE PRNT.
001170     STOP RUN.
001180     END PROGRAM.
```

Giving Total Rounded would subtract Discount from Amount, move the difference to the receiving field Total and round the total to fit in the Picture of the receiving field. If the difference was 12.75 and the Picture for Total 9(2), then the value 13 would be stored in Total. Note that all rounding is done at the .5 level.

You can use the On Size Error option to take special action if the receiving field in an arithmetic operation is not large enough. If this option is not used, the value will simply be truncated and no error message will appear. For example, if you try to move the value 324 into a data item with a Picture of 9(2), the value 24 will be moved into the item and 3 will simply be lost.

Size errors occur under two conditions—when the receiving field is too small to hold the value being moved into it, and when division by zero is attempted. Note the following example: Multiply Amount-1 By Amount-2 Giving Amount-3 On Size Error Perform Error-Routine. This statement instructs the computer to multiply Amount-1 by Amount-2 and move the result into Amount-3. If the data item Amount-3 is not large enough to contain the result, the On Size Error clause is activated, and, in this case, the paragraph named Error-Routine is executed.

Logical Operators

Through the use of the logical operators And and Or, you can test compound conditions. At times it is necessary to perform a particular function only when two or more specific conditions exist. For example, if it is over 85 degrees and the swimming pool is full of water, then go for a swim. This compound condition states that you will only swim if both conditions are true, signified by the word And.

In Cobol the And operator can be used in statements such as If Amount Is Less Than 100 And Budget Is Equal To 75 Display "Adequate Funds". Only when the two conditions are true will the Display statement be executed.

The word Or is used in Cobol in precisely the same way it is used in English. The sentence, "If you have a ticket or $5 then you can get into the movie," means that if you have either the ticket or the money then you can enter. It is necessary to meet only one condition in an Or statement.

A Cobol example of Or is If Name-1 Is Equal To "Mary" Or Name-1 Is Equal To "Sue" Display Phone-Number. If Name-1 contains either name then the contents of the data item

Phone Number will be printed on the screen.

In the examples using And and Or, I used the relational operator Is Equal To. Actually, I could have chosen a number of operators—see Fig. 10.

Final Programming Project

Let's have a final program so we can review the instructions and concepts covered here. The assignment actually consists of two programs—one that accepts data from the keyboard and stores it on the disk, and another that reads in the data and prints a report.

Before the data can be accepted and written to the disk, the file specifications must be made clear. Each record will consist of 44 characters. There will be four fields containing an employee's number, name, budget amount and current expense. The field lengths are as follows: employee number = 5 characters; name = 25 characters; budget amount = 7 characters; and current expense = 7 characters.

The first program should create the disk file, which will be called BudgetFile in the program and BUDGET/SEQ on the disk.

The program should prompt the user to enter the employee number, name, budget amount, and current expense. This data should then be written to the disk and the user asked if he wishes to enter another person's record. If so, ask for the data again and store it on the disk; if not, close the file and end the first program.

Using sample programs, you should find this first program a fairly simple exercise. Although the program is presented in Listing 9, try to write your own version first.

Once the first program is completed, key in the data presented in Fig. 11. Obviously, you will only enter the four field items for each employee. Your second program will calculate the balance.

The second program requires that the data stored on the disk be retrieved and a report printed.

This report should present the employee's number, name, budget amount, current expense and balance. The balance for each employee is calculated by subtracting the current expense from the budget amount—simply use the Cobol Subtract statement. Finally, the report should contain the grand total of each employee's balance. To do this, a running total must be maintained as each record is processed.

The report format is shown in Fig. 11. Note the use of a title and column headings, as well as the final total located at the bottom. Also, the CR (credit) option has been used in the edit data items. Remember that this causes negative values to be accompanied by the letters CR.

[Data-name Literal Arithmetic-expression]	Is {Not}	[< > = Less Than Greater than Equal to]	[Data-name. Literal. Arith-exp.]

Figure 10. Operators

BUDGET REPORT

EMP #	EMPLOYEE NAME	BUDGET AMT	CURRENT EXP	BALANCE
10001	THOMAS, ROY	500.00	250.00	250.00
10002	BOWIE, WENDELL	300.00	450.00	150.00CR
10003	LIPPARD, CINDY	300.00	150.00	150.00
			TOTAL	250.00

Figure 11. Data and Report format

This second program is more complex than the first, but try to write your own before studying the program shown in Listing 9.

Common Cobol Programming Errors

Probably the most common Cobol error is improper spelling and punctuation while keying in the source code. Periods are often omitted and key words misspelled. The simple solution to this is to carefully proofread your program. Those mistakes that you do not find will be flagged by the compiler.

Don't substitute the letter O for the number zero. Note that the zero has a slash through it. Typists must also remember that, to the computer, the letter l is not the same as the number one.

When describing numeric Picture clauses, it is a common mistake to use the . instead of the V. Remember that the decimal point must be represented by a V in the Picture clause of numeric-data items.

Probably one of the most difficult errors to locate occurs when a program line extends beyond column 72. As the program is compiled an error message will be generated but upon proofreading the line no error can be found. Remember that the compiler ignores columns 73 through 80 and does not see any instructions located there. To correct this error, use two program lines for a single instruction.

Despite your efforts, nearly every program that you write will initially contain syntax errors. Your job will be to locate and correct them.

The compiler flags the errors that it finds while compiling the source code and will display them on the screen or printer at your request. Sending error messages to the printer is best—those on the screen quickly scroll past, making it difficult to read them.

Although I have not found a way to make the Model III pause while compiling, the Model II pauses if you depress the space bar. By pausing, you are given time to read the messages on the screen. Press the enter key to resume compiling.

If you have a printer, compile with the full listing going to the printer (use RSCOBOL file name P).

Figure 12 provides a sample of a printout from the compiler, complete with errors.

The compiler flags the location of an error with a dollar sign. It then places another dollar sign where the compilation process resumes. In other words, it jumps over the error and begins compiling again.

Each $ is later followed by a message. The messages are numbered and refer to either the first or second dollar sign. Look at program line number 350 in Fig. 12. Below it are two messages numbered 1 and 2. The first refers to the first $ in line 350, while the second refers to the next $ in the line.

An examination of the two messages shows that a syntax error has been found by the compiler at the start of line 350. What is the error?

Actually, there is no error in line 350. The error occurred in line 340, where a period was omitted. The compiler did not discover the error until it

```
TRS-80 Model II COBOL (RM/COBOL 1.2)                7/21/82  13.50.45   PAGE      1
SOURCE FILE: ERROR                                  OPTION LIST: T P

LINE  DEBUG PG/LN  A...B............................................................

   1        000100 IDENTIFICATION DIVISION.
   2        000110 PROGRAM-ID.                PROJECT2.
   3        000120 AUTHOR.                    student's name.
   4        000130 INSTALLATION.              JTC.
   5        000140 DATE-WRITTEN.              04/21/82.
   6        000150 SECURITY.                  UNCLASSIFIED.
   7        000160
   8        000170 ENVIRONMENT DIVISION.
   9        000180 CONFIGURATION SECTION.
  10        000190 SOURCE-COMPUTER.           MODELII-64K.
  11        000200 OBJECT-COMPUTER.           MODELII-64K.
  12        000210 INPUT-OUTPUT SECTION.
  13        000220 FILE-CONTROL.
  14        000230     SELECT PRNT ASSIGN TO PRINT, "PRINTER".
  15        000240
  16        000250 DATA DIVISION.
  17        000260 FILE SECTION.
  18        000270 FD  PRNT
  19        000280     BLOCK CONTAINS 1 RECORDS
  20        000290     RECORD CONTAINS 132 CHARACTERS
  21        000300     LABEL RECORDS ARE OMITTED
  22        000310     DATA RECORD IS PRINT-LINE.
  23        000320 01  PRINT-LINE                   PICTURE X(132).
  24        000330 WORKING-STORAGE SECTION.
  25        000340 01  ADDRESS-IN-FIELDS
  26        000350     03 NAME-IN                   PICTURE X(30).
                       $                            $
*****   1) SYNTAX    *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E
*****   2) SCAN RESUME  *W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W
  27        000360     03 STREET-IN                 PICTURE X(30).
                       $
*****   1) LEVEL  *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E
  28        000370     03 CITY-STATE-IN             PICTURE X(30).
  29        000380
  30        000390 PROCEDURE DIVISION.
                   $
*****   1) SCAN RESUME  *W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W
  31   >0000 000400 OPEN-FILE-TO-PRINTER.
  32   >0000 000410     OPEN OUTPUT PRNT.
  33   >0008 000420 DISPLAY-AND-ACCEPT-DATA.
  34   >0008 000430     DISPLAY "ENTER NAME:"
  35         000440     LINE 3 POSITION 10 ERASE.
  36   >0012 000450     ACCEPT NAME-IN
                                 $
*****   1) IDENTIFIER   *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E
  37         000460     LINE 3 POSITION 22 PROMPT "." TAB.
                                                     $
*****   1) SCAN RESUME  *W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W
  38   >0014 000470     DISPLAY "ENTER STREET:"
  39         000480     LINE 5 POSITION 8.
TRS-80 Model II COBOL (RM/COBOL 1.2)                7/21/82  13.50.45   PAGE      2
SOURCE FILE: ERROR                                  OPTION LIST: T P

LINE  DEBUG PG/LN  A...B............................................................

  40   >001C 000490     ACCEPT STREET-IN
                                 $
*****   1) IDENTIFIER   *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E
  41         000500     LINE 5 POSITION 22 PROMPT "." TAB.
                                                     $
*****   1) SCAN RESUME  *W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W
  42   >001E 000510     DISPLAY "ENTER CITY & STATE:"
  43         000520     LINE 7 POSITION 2.
  44   >0026 000530     ACCEPT CITY-STATE-IN
                                 $
*****   1) IDENTIFIER   *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E
  45         000540     LINE 7 POSITION 22 PROMPT "." TAB.
                                                     $
*****   1) SCAN RESUME  *W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W
  46   >002A 000550 MOVE-AND-PRINT-DATA.
  47   >002A 000560     MOVE NAME-IN TO PRINT-LINE.
                             $                 $
*****   1) IDENTIFIER   *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E
*****   2) SCAN RESUME  *W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W
  48   >002C 000570     WRITE PRINT-LINE.
  49   >0034 000580     MOVE STREET-IN TO PRINT-LINE.
                             $                   $
*****   1) IDENTIFIER   *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E
*****   2) SCAN RESUME  *W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W
  50   >0036 000590     WRITE PRINT-LINE.
  51   >003E 000600     MOVE CITY-STATE-IN TO PRINT-LINE.
                             $                       $
*****   1) IDENTIFIER   *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E
*****   2) SCAN RESUME  *W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W
  52   >0040 000610     WRITE PRINT-LINE.
  53   >004A 000620 CLOSE-PRINTER-FILE.
  54   >004A 000630     CLOSE PRNT.
  55   >0052 000640 TERMINATE-PROGRAM.
  56   >0052 000650     STOP RUN.
  57         000660     END PROGRAM.
TRS-80 Model II COBOL (RM/COBOL 1.2)                7/21/82  13.50.45   PAGE      3
SOURCE FILE: ERROR                                  OPTION LIST: T P

 ADDRESS   SIZE DEBUG ORDER TYPE                    NAME
            132             FILE                    PRNT
  >0004     132   ANS    0  ALPHANUMERIC             PRINT-LINE


READ ONLY BYTE SIZE =        >00FE

READ/WRITE BYTE SIZE =       >0002

OVERLAY SEGMENT BYTE SIZE = >0000

TOTAL BYTE SIZE =            >01D0

    8 ERRORS

    8 WARNINGS
```

Figure 12. Sample compiler printout.

began compiling line 350.

It is important to understand that the compiler prints an error message when it finds something it doesn't expect. Applying this principle to line 350, we can see that the computer expected to find a period. Since a period must follow line 340, when it moved on to line 350 it was still waiting to find the period. When it found something other than the period, an error message was generated. Quite often, you will find your error in the preceding line, despite the location of the $.

One error can result in many error messages. In the example, the only error is the failure to include a period after the record name in line 340. As a result of this omission, the compiler was unable to recognize the field names; therefore, every time a field name is mentioned in the listing an error message appears. By simply placing a period at the end of line 340, the program will compile with no errors.

With this principle in mind, you should always begin correcting errors from the top of the program. When you reach the point in the listing where you simply cannot find an error despite the $, write the program to the disk and compile it again.

You will find an explanation of the compiler error messages in your TRS-80 Cobol manual in the Use section under the heading Diagnostic Messages. The explanations are not always clear, but as you gain experience in Cobol, they will become easier to understand.

Another set of error codes and messages is presented in the same section under the heading Runtime Diagnostics. These codes refer to errors that occur while the program is actually running—after you type RUNCOBOL file name.

When an error does occur during execution of the program, a code number will be displayed on the screen. Simply locate the corresponding number in your manual for an explanation of the problem. Since we have considered only sequential files, you need only refer to the codes under the heading Sequential Files.

Conclusion

While we haven't covered every aspect of Cobol, you should now know enough of the basics to be intrigued by this language. Get a good textbook, and continue your conquest of Cobol. ■

Sam Perry (210-A Crestview Drive, Smithfield, NC 27577) is an instructor of computer programming and data processing at Johnston Technical College.

The Spelling Bee Is Over

Listen. We're going to let you in on an industry secret: It's not hard to make a good spelling checker.

You see, although spelling checking is new for microcomputers, it's been around on big computers for *years.* And when you get past all the talk, most spelling checkers work the same way. They compare what you've written with a dictionary–and report the errors.

So is there *any* difference? You bet: the dictionary, and the price.

Who Checks The Checker?

The hardest part of a spelling checker to make is the dictionary. It's hard to pick the right words–and spell every one of them perfectly. That's why some popular spelling checkers don't even contain real dictionaries. They use formulas called "hash tables." Which make a hash out of your spelling some of the time.

Other spelling checkers "borrow" their words from printed dictionaries –or copy them from old word lists. Or give a programmer who can't spell "programmer" a chance to write his first dictionary. And as though all this wasn't bad enough, a lot of these companies want to charge you $100, or $200 or even $300!

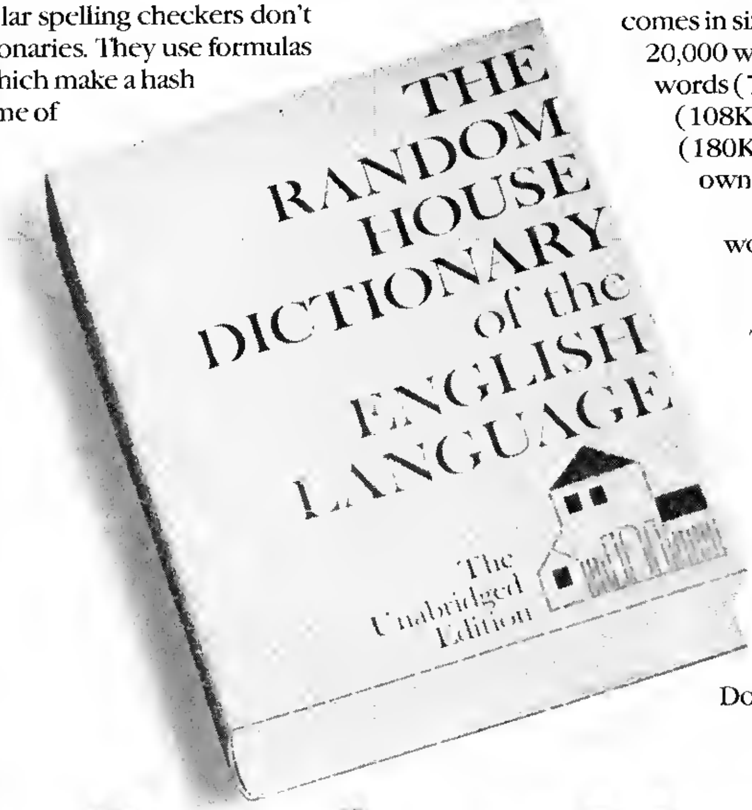
Random House to the Rescue

The Random House ProofReader is based on the famous Random House Dictionary. It is the result of decades of careful work. And it was adapted for computer spelling checking by professional editors, linguists and scholars. So you can depend on it.

Of course, our program is darn good, too. It's fast. It doesn't take up too much disk space. It shows you the error and the sentence it's in. It lets you correct the mistake without reediting. And if you don't know how to spell a word, it suggests the spelling. It even corrects your correction.

The Random House ProofReader comes in sizes to fit all systems: 20,000 words (55K), 32,000 words (74K), 50,000 words (108K) and 83,000 words (180K). You can add your own words with a single keystroke. And it works with all CP/M®, IBM Personal Computer®, and TRS-80® Model I/III word processors.*

How much does the Random House ProofReader cost? Only $50.00. Why do we sell it for $50 when the others want up to $300? Don't ask us. Ask them.



The Random House ProofReader $50

✓65

For orders or information, call 505-281-1634. VISA, Master Card accepted. Or write Random House ProofReader, Box 339-M, Tijeras, New Mexico 87059. Please enclose $50 and specify your computer model, disk size and memory. Dealer inquiries invited.

* Random House and the House design are registered trademarks of Random House Inc. CP/M is a registered trademark of Digital Research, Inc. TRS-80 is a trademark of Tandy Corporation. IBM and IBM Personal Computer are registered trademarks of International Business Machines Corp. Dictionary copyright © 1982 Random House, Inc. Program and advertisement copyright © 1982 Aspen Software, Inc. The TRS-80 versions do not suggest correct spellings.

Disk Tiny Pascal

by David M. Silver

Did you have to abandon all your cassette-based Tiny Pascal programs when you upgraded to a disk system? David's got the answer.

Do you want to modify Radio Shack's cassette-based Tiny Pascal for disk? This article carries on from where "Modify Tiny Pascal for Disk" left off (*80 Micro*, July 81).

These machine-language programs will allow you to save and load source and compiled p-code programs using the disk. The cassette routines are still present so you can store all your old cassette programs on disk.

An additional routine prints source programs, and a command returns control to DOS. You can even prepare source programs with word processors such as Scripsit or Electric Pencil.

Tape to Disk

The original 32K program loads into addresses 4D90H–73EFH. Upon execution, a block move transfers 900H bytes from 4D90H down to 4090H. According to the Tiny Pascal manual, addresses from 4060H upward are used, although I found no references to addresses below 4090H. However, DOS uses memory up to 51FFH. In order to use the DOS routines for disk I/O, the Tiny Pascal program must be relocated above the DOS area. We will shift Tiny Pascal by 1200H bytes, so it occupies memory from 5290H upward.

There are many ways to save the cassette program on disk. NEWDOS users can use the LMOFFSET program. The new load point address should be 4D90H plus 1200H or 5F90H. Suppress the appendage, and call the disk program PASCAL/CMD.

Other programs, such as RSM-2D, perform the same function.

A short machine-language program (see Listing 1) can also perform the job. Using Debug, type the hex code into 9000H–900DH. Then reset the computer while holding the break key down. Answer the memory-size prompt with enter. Enter System, followed by PAS32K, and load the 32K Tiny Pascal from cassette. At the next prompt type /36864. (This is the address of the block-move routine that is still in memory.)

The routine moves the program to an area of memory above DOS and then reboots the computer, this time with disks. Now enter the following command:

DUMP PASCAL/CMD (START=X'5F90', END=X'85EF')

Tiny Pascal is now on disk, but not in an executable form. Addresses in the program still refer to the original program locations. We must change these references. (At this stage, make a backup of this disk file in case you make errors in the following steps.)

Completing Relocation

Moving the program code in memory was relatively simple. In the second stage we must offset all references to addresses 4090H–73EFH by 1200H by adding 12H to the most significant byte of these address references. Table 1 lists all the addresses of bytes that must be altered, as well as the old and new contents of each byte. Change these by using Debug.

I found other changes necessary. Because the system often rebooted or hung up, I disabled the interrupts.

The break key was no longer functional because Level II ROM jumps to 400CH when the break key is pressed. A jump vector is placed here in DOS, whereas for Level II operation, there is just a return opcode. (I found that this was the only Level II environment expected by Tiny Pascal.) I used a return opcode (C9H) to replace the jump opcode (C3H) at 400CH.

Cassettes created by Tiny Pascal include the address of the data to be loaded as part of each data block. The address bytes are also used in the checksum calculation. This meant that cassettes from the original Tiny Pascal would load into the original, lower area of memory and bomb the system. Since we always load data starting

The Key Box

Model I
32K RAM
Tiny Pascal
One Disk Drive

from a known location (8900H for disk Tiny Pascal), the address information is unnecessary. I modified the cassette routines to accept cassettes created by both the original cassette version of Tiny Pascal and the disk version described in this article. (I doubt that you'll want to save programs on cassette anyway.) The new cassette routines will also read cassettes created with the 16K version of Tiny Pascal.

Table 1 also lists the changes necessary to solve these problems. Use Debug and work methodically to avoid errors. If you change the wrong byte by mistake, the back-up copy of PASCAL/CMD may rescue you. Note that some locations are shown with xx as the original contents. These are in buffer areas in the original Tiny Pascal, and the contents may vary from copy to copy.

Exit from Debug and enter the following command:

```
DUMP PASCAL/CMD (START=X'5F8A',
     END=X'85EF',TRA=X'5F8A')
```

Now test all your work to this stage; the program should perform identically to the original cassette version. If you have problems, check all your modifications with Table 1. Ensure that you have made the correct changes for the memory size of your system.

Determining the Bytes to be Changed

The Tiny Pascal manual gives a memory map of the system. From this I determined that the bulk of the changes would be made to the interpreter, I/O and runtime routines, and the entry-points table. The routines are written in machine language and the entry-points table refers to machine addresses. The remainder of the package is p-code for the editor and compiler. P-code is relocatable; modifications are only required if the p-code refers to an absolute memory address. In this case, a few references are made to the system control block in the editor and compiler p-code.

Using a disassembler and the memory map provided, I disassembled the original code from 4090H–498FH. (The code is placed there by the initial block move from 4D90H.) This is the only machine-language (not p-code) part of the system. The result was a heap of about 25 pages of Assembly code! All references to addresses in the range 4090H–73EFH would have to be changed for relocation.

The most common instructions to be changed were jumps and calls, and loads to and from absolute addresses. Relative jumps do not have to be altered. No indexed addressing is used in the program, so all the references to addresses were listed by the disassembler. An exception was the most significant byte of the address of the entry points table. This was used as data in one instruction and had to be altered. I also noted references to the location of the text buffer, editor p-code, compiler p-code, and run-time stack, as well as the memory size of the system.

Another problem is distinguishing between data and program code. This is partly a matter of experience. Strange hex code is worth converting to ASCII: The resulting data is often a message. The area of strange code could also be used as a buffer.

The source listing of the compiler indicated the references to the system control block in the compiler p-code. Since the address of the control block was defined as a constant in the listing, I expected to find the same value in the p-code. Similarly, expressions that used this constant would also need to have their corresponding p-code altered. I searched the p-code areas for references in the range 4180H–419BH, and noted the addresses.

Adding the Extensions to Tiny Pascal

The new commands with the extensions to Tiny Pascal are shown in Table 2. I used the original cassette

Table 1

```
(Note that all address and byte values are in hex)
Change bytes at these addresses from 40 to 52: (relocation)
5F95  663C  663F  6645  6648  666E  6699  66D2  6729  6738
673E  6784  678D  67BF  67CD  67FE  6804  6808  6822  6833
683A  6862  6882

Change bytes at these addresses from 41 to 53: (relocation)
60A2  60A5  60A8  60B4  60E4  60F7  60FD  6107  611C  611F
6123  6142  6177  6238  626C  627F  6284  6287  628A  6376
656E  6669  6680  6683  6686  6689  668C  6690  6694  669F
66A9  66BE  66C1  66C4  66CD  66D8  66DB  66DE  66F0  66F4
66FD  6701  6714  671C  6722  672F  6748  6756  6759  6799
67C7  67CA  6E35  6E3C  6E50  7086  71C8  71D0  71DB  71E4
75E5  859A

Change bytes at these addresses from 42 to 54: (relocation)
6001  6003  6005  6007  6009  600B  600D  600F  6011  6017
6019  601B  601D  601F  602B  602D  6101  6135  6146  6166
6170  617C  6182  61A3  61A6  61BC  61BF  61D2  61D6  61D9
61EE  61F1  6211  623C  6346  634C  637D  63A2  655F

Change bytes at these addresses from 43 to 55: (relocation)
6013  6015  6021  6023  6025  6027  6029  605B  605D  605F
6217  6275  6293  629D  62AF  62CB  62CE  62E3  6311  6326
6388  6524  6535  6558  6562

Change bytes at these addresses from 44 to 56: (relocation)
602F  6031  6033  6035  6037  6039  6061  60FA  617F  6260
6268  62DB  62E0  62E6  6304  630C  6314  632A  6380  6390
639A  63A5  63A8  63AB  63B2  63B6  63CF  63D9  63DD  63E1
63E4  63EC  63EF  63F2  6410  6416  643C  644D  6456  645C
6471  64A5  64B0  64C4  64D5  64D8  64EB  6538  65B5  65F8
6600  660B  660F  661A  661E

Change bytes at these addresses from 45 to 57: (relocation)
603B  603D  603F  6041  6043  6045  6047  6049  604B  604D
604F  6051  6053  6055  6057  631A  63E8  6431  6451  646A
647E  6496  649C  64A2  64C9  64CC  64DF  65BC

Change bytes at these addresses from 46 to 58: (relocation)
6059  6063  6065  6067  6069  606B  606D  606F  6071  6073
6075  60E8  6149  61AA  61C3  61DD  61F5  621B  6225  6228
6298  62AC  62D7  632F  6337  650D  6511  6517  651F  6543
```

Table 1 continues

Table 1 continued

```
654A  658E  65A3  65C3

Change bytes at these addresses from 47 to 59: (relocation)
5F9D  6077  6095  60B7  665B  66F7  6704  6735  6744  679C

Change bytes at these addresses from 48 to 5A: (relocation)
6642  664B  664E  6660  6665  6671  6675  667C  669C  66A3
66A6  66B5  66C9  66E1  66EA  66ED  66FA  6707  6710  673B
6741  6791  679F  67A9  67B0  67B8  67C2  6825

Change bytes at these addresses from 49 to 5B: (relocation)
608F  6732  6828

Change at 5F8A to 5F8F:  (enable Break key & disable
from xx xx xx xx xx xx     interrupts)
to   3E C9 32 0C 40 F3

Change bytes at these addresses
from 4D to 5F:  5F92      (relocation of initial block move)
from F0 to 00:  6080  608C  (relocation of source buffer)
from 73 to 89:  6081  608D     "      "     "      "
from 56 to 68:  6089      (relocation of editor p-code)
from 5F to 71:  608B      (relocation of compiler p-code)
from BF to FF:  (for 48k systems only)  6091  6093 (memory
                                                     size)
from 0C to 23:  676D  6773  (space for DOS DCB + 3)
from F2 to 56:  6750  677E  (relocation of command buffer)
from 40 to 88:  6751  677F  (    "      "     "      "   )
from 12 to F2:  67CF  (remainder of the changes are
from 02 to 52:  67D0   patches for the cassette loader)
from 22 to CD:  6802
from 98 to F9:  6803
from 35 to CC:  680D
from 02 to 53:  680E
from 77 to 00:  680F
from 2B to 1B:  682F
from EB to 00:  6830

Change at 5FF2 to 5FFD
from xx xx xx xx xx xx xx xx xx xx xx xx
to   CD 12 02 11 00 89 C9 ED 53 98 52 C9

Change at 60CC to 60D1:
from xx xx xx xx xx xx
to   CD 35 02 12 13 C9
```

```
Change bytes at these addresses
from 4C to 47:  6658  (change 'L' to 'G' for cassette load)
from 57 to 53:  665D  (change 'W' to 'S' for cassette save)
from 95 to F0:  6664  (patch extension to monitor)
from 5A to 85:  6665  (  "        "       "     "   )
```

Table 3

commands for the equivalent disk operations because I still tended to type LS when I wanted to load a source file from disk. Since I expected to use the cassette routines rarely after the initial rush to store cassette programs on disk, changing the names of the cassette commands caused no inconvenience.

To access these extensions to Tiny Pascal, the monitor must recognize the new commands and jump to the required routine. Program Listing 2 is the Assembly code for the extended monitor and routines. The comments with the program describe its operation.

For the disk operations, the return opcode at 400CH is replaced by a jump; this is restored upon completion of disk I/O to reenable the break key. I have used the standard DOS Open and INIT calls, as well as the write-byte-to-disk (Call 1BH)) and read-byte-from-disk (Call 13H)) routines.

Because p-code can contain bytes of any value, an EOF byte cannot be defined. Thus, my routines store the byte count of the p-code program as the first two bytes of the file, followed by the p-code. The byte count information is then used by the loader to determine the number of bytes to read.

Add the hex code to PASCAL/CMD by using Debug or creating the file using EDTASM. First, load the PASCAL/CMD file you created previously. Then enter Debug and type the hex code into locations 85F0H-8755H, or load the assembled EDTASM object code. Finally, patch these new routines into PASCAL/CMD and change the names of the cassette commands. Table 3 lists the addresses of the four bytes to be changed, along with the old and new values.

Then enter the following command:

DUMP PASCAL/CMD (START=X'5F8A',
END=X'8755'.TRA=X'5F8A')

Command	Function
SS filespec	Save source program on cassette (same as old WS)
SP filespec	Save P-code on cassette (same as old WP)
GS filespec	Get source program from cassette (same as old LS)
GP filespec	Get p-code from cassette (same as old LP)
WS filespec	Save source program on disk
WP filespec	Save p-code on disk
LS filespec	Load source program from disk
LP filespec	Load p-code from disk
P	Print source program
X	Return to DOS

Table 2

```
                00100 ;
                00110 ;
                00120 ;
                00130 ;
                00140 ;
                00150 ;********* PROGRAM TO MOVE TINY PASCAL **********
                00160 ;
                00170 ; WRITTEN BY D.M. SILVER  12TH NOV. 1981
                00180 ; 15 HOWARD ST. KEW 3101. VIC. AUSTRALIA
                00190 ;
9000            00200        ORG     9000H
9000 21EF73     00210 MOVE   LD      HL,73EFH           ;END OF TINY PASCAL
9003 11EF85     00220        LD      DE,85EFH           ;DESTINATION
9006 016026     00230        LD      BC,2660H           ;BYTE COUNT
9009 EDB8       00240        LDDR                       ;MOVE IT
900B C30000     00250        JP      0                  ;REBOOT SYSTEM
9000            00260        END     MOVE
00000 TOTAL ERRORS
```

Program Listing 1

Program Listing 2

```
                00100 ;
                00110 ;
                00120 ;
                00130 ;
                00140 ;
                00150 ;********** EXTENSIONS FOR DISK TINY PASCAL *********
                00160
                00170 ; WRITTEN BY D.M. SILVER  12TH NOV. 1981
                00180 ; 15 HOWARD ST. KEW 3101. VIC. AUSTRALIA
                00190 ;
                00200 ;************** CONTINUATION OF MONITOR *************
                00210
85F0            00220        ORG     85F0H
85F0 FE58       00230 MONCTD CP      'X'          ;TEST FOR RETURN TO DOS COMMAND
85F2 CA4E87     00240        JP      Z,EXIT
85F5 FE50       00250        CP      'P'          ;TEST FOR PRINT COMMAND
```

Listing 2 continued

You should now have a fully operational disk version of Tiny Pascal. The disk I/O routines incorporate error-checking and display the conventional DOS error messages. Note that there must be only one space between a disk command (such as LS) and the file name. You can stop the print routine at any time by pressing the break key. This allows you to escape if you accidentally try to print a long file or if you don't have a printer connected. That's right—no more computer hangups!

I have tried this Tiny Pascal with TRSDOS 2.3, NEWDOS 2.1, and NEWDOS80. When a disk error occurs in TRSDOS, the message displayed is followed by meaningless graphics that scroll the message off the screen. However, control still returns to the monitor. I don't know why this occurs since I used a documented TRSDOS call.

Finally, the routine that loads source code from disk is compatible with word processors such as Scripsit and Electric Pencil. The routine accepts the ASCII characters between 20H and 7FH, carriage return (0DH), and the Tiny Pascal editor tab character (09H). Any other hex code is interpreted as the EOF byte. Scripsit users should save the files with the ASCII option. Deleting to the end of text after the last character is also a sensible precaution to avoid garbage after the last valid character (which should be an end-of-line marker). Also, the only control character used should be the end-of-line marker—include no formatting instructions!

Disk Tiny Pascal is quite a powerful package for applications requiring only integer arithmetic. Now all you disk users can publish programs written in Tiny Pascal! ■

```
85F7 CA1787           JP     Z,PRINT
85FA FE4C             CP     'L'          ;TEST FOR DISK LOAD COMMAND
85FC C20F86           JP     NZ,TESTW
85FF CD825A           CALL   5A82H        ;GET CHARACTER FOLLLOWING 'L'
8602 FE53             CP     'S'          ;TEST FOR LOAD SOURCE COMMAND
8604 CA9786           JP     Z,LS
8607 FE50             CP     'P'          ;TEST FOR LOAD P-CODE COMMAND
8609 CAC686           JP     Z,LP
860C C3955A           JP     5A95H        ;BAD COMMAND, RETURN TO MONITOR
860F FE57      TESTW  CP     'W'          ;TEST FOR WRITE TO DISK COMMAND
8611 C2955A           JP     NZ,5A95H     ;BAD COMMAND, RETURN TO MONITOR
8614 CD825A           CALL   5A82H        ;GET CHARACTER FOLLOWING 'W'
8617 FE53             CP     'S'          ;TEST FOR WRITE SOURCE CODE
8619 CA2486           JP     Z,WS
861C FE50             CP     'P'          ;TEST FOR WRITE P-CODE COMMAND
861E CA5A86           JP     Z,WP
8621 C3955A           JP     5A95H        ;BAD COMMAND, RETURN TO MONITOR
               ;*************** WRITE SOURCE CODE TO DISK **************
8624 3A8353    WS     LD     A,(5383H)    ;TEST FOR VALID SOURCE
8627 B7               OR     A
8628 CA365A           JP     Z,5A36H
862B CD0987           CALL   SETUP        ;SET UP DOS CALL AND VECTOR
862E CD2044           CALL   4420H        ;CALL DOS INIT & TEST FOR ERROR
8631 C2FB86           JP     NZ,ERROR
8634 2A8053           LD     HL,(5380H)   ;LOAD POINTER TO SOURCE CODE
8637 7E        LOOP1  LD     A,(HL)       ;GET TEXT BYTE FROM BUFFER
8638 FEFF             CP     0FFH         ;TEST FOR END-OF-FILE
863A 2809             JR     Z,EOF1
863C CD1B00           CALL   1BH          ;WRITE BYTE TO DISK
863F C2FB86           JP     NZ,ERROR
8642 23               INC    HL           ;UPDATE POINTER TO SOURCE CODE
8643 18F2             JR     LOOP1        ;LOOP FOR NEXT BYTE
8645 CD1B00    EOF1   CALL   1BH          ;WRITE EOF BYTE (FF) TO DISK
8648 C2FB86           JP     NZ,ERROR
864B CD2844    FINISH CALL   4428H        ;CLOSE FILE
864E C2FB86           JP     NZ,ERROR
8651 F3               DI                  ;DISABLE INTERRUPTS
8652 3EC9             LD     A,0C9H       ;RESTORE BREAK KEY OPERATION
8654 320C40           LD     (400CH),A
8657 C34359           JP     5943H        ;RETURN TO MONITOR
               ;*************** WRITE P-CODE TO DISK **************
865A 3A8753    WP     LD     A,(5387H)    ;TEST FOR VALID P-CODE
865D B7               OR     A
865E CA275A           JP     Z,5A27H
8661 CD0987           CALL   SETUP        ;SET UP DOS DCB AND VECTOR
8664 CD2044           CALL   4420H        ;CALL DOS INIT
8667 C2FB86           JP     NZ,ERROR
866A 2A8653           LD     HL,(5386H)   ;CALCULATE LENGTH OF P-CODE
866D ED4B8453         LD     BC,(5384H)
8671 AF               XOR    A
8672 ED42             SBC    HL,BC
8674 23               INC    HL
8675 4D               LD     C,L          ;STORE BYTE COUNT IN BC REG
8676 44               LD     B,H
8677 79               LD     A,C
8678 CD1B00           CALL   1BH          ;WRITE BYTE COUNT LSB TO DISK
867B C2FB86           JP     NZ,ERROR
867E 78               LD     A,B
867F CD1B00           CALL   1BH          ;WRITE BYTE COUNT MSB TO DISK
8682 C2FB86           JP     NZ,ERROR
8685 2A8453           LD     HL,(5384H)   ;LOAD POINTER TO P-CODE
8688 7E        LOOP2  LD     A,(HL)       ;GET BYTE OF P-CODE
8689 CD1B00           CALL   1BH          ;WRITE BYTE TO DISK
868C C2FB86           JP     NZ,ERROR
868F 23               INC    HL           ;UPDATE POINTER TO P-CODE
8690 0B               DEC    BC           ;UPDATE BYTE COUNT
8691 79               LD     A,C
8692 B0               OR     B
8693 20F3             JR     NZ,LOOP2     ;LOOP IF MORE BYTES TO WRITE
8695 18B4             JR     FINISH       ;CLOSE FILE, ETC.
               ;************** LOAD SOURCE CODE FROM DISK **************
8697 CD0987    LS     CALL   SETUP        ;SET UP DOS DCB AND VECTOR
869A CD2444           CALL   4424H        ;CALL DOS OPEN
869D C2FB86           JP     NZ,ERROR
86A0 2A8053           LD     HL,(5380H)   ;LOAD POINTER TO SOURCE CODE
86A3 CD1300    LOOP3  CALL   13H          ;READ BYTE FROM DISK
86A6 C2FB86           JP     NZ,ERROR
86A9 FE0D             CP     0DH          ;TEST FOR VALID CHAR - MUST BE
86AB 280E             JR     Z,OK         ;09H, 0DH, OR BETWEEN 20H & 7FH
86AD FE09             CP     09H          ;INCLUSIVE
86AF 280A             JR     Z,OK
86B1 FE20             CP     20H
86B3 FABF86           JP     M,EOF2       ;IF NOT VALID CHAR, MUST BE EOF
86B6 FE7F             CP     7FH
86B8 F2BF86           JP     P,EOF2
86BB 77        OK     LD     (HL),A       ;STORE BYTE IN TEXT BUFFER
```

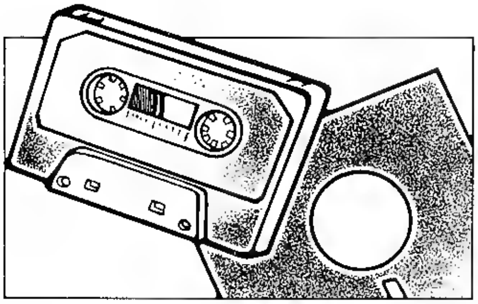
Listing 2 continues

Listing 2 continued

```
86BC 23                     INC     HL          ;UPDATE POINTER TO SOURCE CODE
86BD 18E4                   JR      LOOP3       ;LOOP FOR NEXT CHARACTER
86BF 36FF         EOF2      LD      (HL),ØFFH   ;STORE EOF BYTE IN TEXT BUFFER
86C1 228253                 LD      (5382H),HL  ;STORE END OF SOURCE ADDRESS
86C4 1885                   JR      FINISH      ;CLOSE FILE, ETC.


                  ;*************** LOAD P-CODE FROM DISK **************

86C6 CDØ987       LP        CALL    SETUP       ;SET UP DOS DCB AND VECTOR
86C9 CD2444                 CALL    4424H       ;CALL DOS OPEN
86CC C2FB86                 JP      NZ,ERROR
86CF 2A8Ø53                 LD      HL,(538ØH)  ;LOAD POINTER TO P-CODE
86D2 228453                 LD      (5384H),HL  ;STORE START OF P-CODE ADDRESS
86D5 CD13ØØ                 CALL    13H         ;READ LSB OF BYTE COUNT FROM DISK

86D8 C2FB86                 JP      NZ,ERROR
86DB 4F                     LD      C,A
86DC CD13ØØ                 CALL    13H         ;READ MSB OF BYTE COUNT FROM DISK

86DF C2FB86                 JP      NZ,ERROR
86E2 47                     LD      B,A         ;BYTE COUNT IS IN BC REG
86E3 CD13ØØ       LOOP4     CALL    13H         ;READ BYTE OF P-CODE FROM DISK
86E6 C2FB86                 JP      NZ,ERROR
86E9 77                     LD      (HL),A      ;STORE P-CODE IN BUFFER
86EA 23                     INC     HL          ;UPDATE POINTER TO P-CODE
86EB ØB                     DEC     BC          ;DECREMENT BYTE COUNT
86EC 79                     LD      A,C
86ED BØ                     OR      B
86EE 2ØF3                   JR      NZ,LOOP4    ;LOOP IF MORE BYTES TO BE READ
86FØ 2B                     DEC     HL
86F1 228653                 LD      (5386H),HL  ;STORE END OF P-CODE ADDRESS
86F4 AF                     XOR     A
86F5 328353                 LD      (5383H),A   ;ZERO END OF SOURCE ADDRESS
86F8 C34B86                 JP      FINISH      ;CLOSE FILE, ETC.


                  ;************* DISK ERROR ROUTINE **************

86FB F68Ø         ERROR     OR      8ØH         ;SET UP DOS ERROR CODE
86FD CDØ944                 CALL    44Ø9H       ;DISPLAY DOS ERROR MESSAGE
87ØØ 3EC9                   LD      A,ØC9H      ;RESTORE BREAK KEY OPERATION
87Ø2 32ØC4Ø                 LD      (4ØØCH),A
87Ø5 F3                     DI                  ;DISABLE INTERRUPTS
87Ø6 C34359                 JP      5943H       ;RETURN TO MONITOR


                  ;*************** SET UP DCB FOR DOS CALL **************

87Ø9 115988       SETUP     LD      DE,DCB+3    ;ADDRESS OF FILE NAME
87ØC 215687                 LD      HL,BUFFER   ;ADDRESS OF BUFFER FOR DISK I/O
87ØF Ø6ØØ                   LD      B,Ø         ;LRL=256 BYTES
8711 3EC3                   LD      A,ØC3H      ;SET UP DOS VECTOR
8713 32ØC4Ø                 LD      (4ØØCH),A
8716 C9                     RET


                  ;****************** PRINT ROUTINE ******************

8717 3A8353       PRINT     LD      A,(5383H)   ;TEST FOR VALID SOURCE FILE
871A B7                     OR      A
871B CA365A                 JP      Z,5A36H
871E 2A8Ø53                 LD      HL,(538ØH)  ;LOAD POINTER FOR SOURCE CODE
8721 7E           PLOOP     LD      A,(HL)      ;GET BYTE OF TEXT
8722 FEFF                   CP      ØFFH        ;TEST FOR END OF FILE
8724 CA4359                 JP      Z,5943H     ;RETURN TO MONITOR WHEN FINISHED
8727 FEØ9                   CP      9           ;TEST FOR TAB CHARACTER
8729 2ØØC                   JR      NZ,P1
872B 3E2Ø                   LD      A,2ØH       ;IF TAB, SUBSTITUTE WITH 3 SPACES
872D CD3D87                 CALL    PCHAR       ;PRINT SPACE
873Ø 3E2Ø                   LD      A,2ØH
8732 CD3D87                 CALL    PCHAR       ;PRINT SPACE
8735 3E2Ø                   LD      A,2ØH
8737 CD3D87       P1        CALL    PCHAR       ;PRINT CHARACTER
873A 23                     INC     HL          ;UPDATE POINTER TO SOURCE CODE
873B 18E4                   JR      PLOOP       ;LOOP TO PRINT NEXT CHARACTER
873D 32E837       PCHAR     LD      (37E8H),A   ;PRINT CHARACTER
874Ø CDC453       WAIT      CALL    53C4H       ;CALL INKEY ROUTINE
8743 CA4359                 JP      Z,5943H     ;IF BREAK KEY, RETURN TO MONITOR
8746 3AE837                 LD      A,(37E8H)   ;GET PRINTER STATUS
8749 CB7F                   BIT     7,A
874B 2ØF3                   JR      NZ,WAIT     ;LOOP WHILE BUSY
874D C9                     RET


                  ;**************** RETURN TO DOS ****************

874E 3EC3         EXIT      LD      A,ØC3H      ;RESTORE DOS VECTOR
875Ø 32ØC4Ø                 LD      (4ØØCH),A
8753 C32D4Ø                 JP      4Ø2DH       ;JUMP TO DOS


                  ;**************** BUFFER AREAS ****************

Ø1ØØ              BUFFER    DEFS    256         ;BUFFER FOR DOS I/O
ØØ23              DCB       DEFS    35          ;COMMAND BUFFER AND DOS DCB
ØØØØ                        END
ØØØØØ TOTAL ERRORS
```

RELOAD 80

by Art Huston



Index and guide to Load 80

Charter subscribers to *80 Micro* remember the dark days...long hours spent typing in a program only to have it crash with a typing error. Since April 1981, Load 80 has been making such aggravation a thing of the past.

An ad in the April 1981 magazine introduced Load 80 as "a monthly dump of the major program listings in *80 Microcomputing* on cassette."

This column began in August 1981 as technical advice to Load 80 users, and in March 1982 Load 80 expanded to include disks. Originally conceived for the Model I/III only, Load 80 begins coverage of the Color Computer in December 1982, and now has plans to appear on a major networking service.

This Anniversary Reload 80 column is a complete guide to using Load 80.

Model I and III Load 80

Tapes for the Model I and III are available monthly for \$9.97 or at a yearly subscription rate of \$99.97. Each tape is 30 minutes long and recorded at 500 baud (the low rate for the Model III). The tapes usually include 12 or more programs totalling up to 100K bytes.

Disks for the Model I and III are \$19.97 each or \$199.97 for a year's subscription. The disks are Model I, single-density, 35-track data disks. They are ready to use in a Model I and in a Model III after using the TRSDOS Convert utility (see details below). The disk is two sided and can be flipped to read both sides. As with tapes, the disks include 12 or more programs.

A special Anniversary Load 80 is available for the Model I and III on both tape and disk for \$9.97 and \$19.97. Articles with programs included in Load 80 are marked with the Load 80 logo in the index and on the first page of the article. A complete list is also printed below in the Load 80 directory.

Back issues of all Load 80s are available. Some of the programs published in the last year include two machine-language monitors in April, a machine-language disk zapper in June/July, 23 games in August's game issue, and a Basic Compiler in October. A complete listing of all programs ever included on Load 80 is published in this special issue of *80 Micro*.

Color Computer Load 80 Cassettes

The first Color Computer Load 80 is available starting with the December 1982 *80 Micro*. This cassette version of the major 1982 Color Computer program listings is available for only \$14.97.

Subsequent Color Load 80s will be available every three months covering the past three issues of the magazine for only \$9.97. At the Color Computer's 1,500-baud rate, these 15-minute tapes are real bargains. Color Load 80 is not available for this special issue.

Documentation

The Load 80 documentation consists of a list of the programs, the language each is in, and a phone number and name to call if you need help. The documentation for the individual programs are the accompanying articles in *80 Micro*. If you don't read it you cannot expect to use Load 80.

Each program is the program listing as it appears in the magazine. Some programs include remark statements detailing changes needed to run under different systems. Your guide, however, should be the keybox on the first page of each article. If your system is not listed there, the program will probably not run correctly on it.

This column is also a valuable source of Load 80 information, on everything from loading tapes to modifying programs to run under different systems.

Model I/III Cassettes

Model I/III cassettes are written at 500 baud. Model III users must respond L to the CASS? prompt or type POKE 16913,0 to set the baud rate low. On the Model I with Disk Basic, type CMD"T" before loading any files.

Basic files are loaded by using CLOAD"file name." If you use disks, a simple SAVE"file name" will save the program to disk.

Loading tapes requires changing the volume on the recorder. Start at one end of the volume scale and work up if

Program	Title	Page	Comments
1	COPYRGHT/BAS	—	None
2	XASMZ865/BAS	130	None
3	STARTRCK/BAS	156	None
4	CASSREAD/BAS	164	None
5	CASSREAD/SRC	164	Needs EDTASM
6	GALLERY/BAS	190	None
7	SENTENCE/BAS	228	None
8	DSKCLEAN/BAS	238	None
9	SPCBOMBR/SRC	244	Needs EDTASM
10	DVMUSIC/BAS	253	None
11	DVMUSIC/SRC	253	Needs EDTASM
12	FLEXCAT1/BAS	264	None
13	FLEXCAT2/BAS	264	None
14	FLEXCAT3/BAS	264	None
15	CATALOGS/BAS	316	None
16	PROGRAMA/BAS	316	None
17	PROGRAMB/BAS	316	None

Anniversary Load 80 Directory

you have trouble. Other measures that might help are aligning the heads of the recorder (see September 1981 Reload 80), demagnetizing the heads, or cleaning the heads.

Model I/III Disks

Load 80 disks are supplied on TDOS (TinyDOS), the DOSPLUS 3.4 kernel. These disks are flippies, meaning that each side has data on it. To read the other side, simply flip the disk over.

Model I users can simply insert the disk in drive 0. Model III users must use Convert to copy the files to TRSDOS:

- Back up a copy of TRSDOS. Insert the back-up in 0 and remove the other disk.
- Type Purge and then the master password for the back-up disk.
- Type Y for each file. It will be purged from the back-up disk.
- Insert the Load 80 disk in drive 1. Type Convert.
- Answer source as 1, destination as 0. The Load 80 files will be copied from the Load 80 disk onto the TRSDOS disk.
- Flip the Load 80 disk and repeat the last two steps.

All Load 80 files are now on your TRSDOS disk, ready for use.

Basic files are loaded by entering Disk Basic and typing LOAD"file name." Remember that if a program is written for a cassette-based machine, it might not run on your system.

Using Assembly-Language Files

Many utilities and high-speed games are written in machine language. These programs are a numeric code immediately recognizable to the computer. For instance, the sequence of numbers 33,0,60,54,65 means "place a 65 in memory address 15360."

Machine-language programs are written using Assembly language, which uses more English-like instructions than machine language. For example:

```
LD HL,15360
LD (HL),65
```

means the same as the sequence of numbers above. Instructions like this are called source code and must be assembled into machine language before being executed.

So programmers can have the option

of modifying the program, Load 80 sells programs in Assembly language, not machine language. This requires that you own an editor/assembler, a program that writes (edits) Assembly language and assembles it. Two such programs are EDTASM from Radio Shack and EDTASM-Plus from Apparat. Not all editor/assemblers will load files from both tape and disk, so be sure you buy the right one.

Radio Shack introduced their disk version of EDTASM later than many other companies and made their source code incompatible. For this reason Load 80 uses the Apparat format for source code. If you are buying Load 80 disks be sure to buy an editor/assembler that uses the same format.

Basic programs do not, of course, require an editor/assembler. ■

Load 80 Index, 1981–1982

Month	Year	Program Name	Page Number
April	1981	COMPARE	271
		EDUC4	144
		EDUC8	144
		CENTS	192
		DOTGAM	210
		CHECKS	214
		WHERZIT1	252
		WHERZIT2	252
		EXPO	266
		BOOTHE4	117
		BOOTHE5	117
		BOOTHE6	117
		BOOTHE7	117
		BOOTHE/SRC	117
		SIMUL80	154
		ASTROL	179
		MANAGE	186
May	1981	VARDOC	291
		SORT	264
		HOMECALC	100
		ENCODE/BAS	234
		DISCO	195
		BOA	294
		DOME	128
		CARDEAL	113
		RANDOM	246
		WRAP	109
		INVID	176
		DOLLAR1	216
		DOLLAR2	218
		AVERAGE	292
		SCROLLER	204
		CARAN	93
		SCROLLER/SRC	205
		LIST1/SRC	39
		LIST3/SRC	40
		SHFTRV/SRC	262
		INVERT/SRC	176
		COMPRS/SRC	270
		LINE1/SRC	139
		RAMTST/SRC	148
		SARGIO1/SRC	274
		SARGIO2/SRC	274
June	1981	BOOTHE3	68
		MENMGR/BAS	178
		SUB	186
		COPYKILL	224
		MILEAG	236

Month	Year	Program Name	Page Number	
		LUNAR	244	
		P/CMD	246	
		ELECTR	288	
		FORECAST/MKT	269	
		STOCK/ANA	274	
		FIN/ANA	284	
		LEDGER	222	(Pub. May 1981)
July	1981	PILOT	122	
		PILOT2	124	
		BBOX	217	
		DISAS	244	
		TPASCAL	154	
		TWOHAF	202	
		TWOHAF2	202	
		HISPEED	280	
August	1981	FRKNSTN	92	
		TANK	104	
		SPCEMP	106	
		STRGURD	116	
		RACE	122	
		EPEXCR	126	
		EPSLCT	126	
		MORSE	150	
		MESSAGE	150	
		DATGEN/SRC	168	
		VIDSPC	180	
		RELSHP	192	
		VCESYN	208	
		SRTSOFSR	214	
		DISAS	240	
		PHODEX	248	
		SPRUSR/SRC	256	
		DIRPROG	270	
		STRTRK	86	
		CMDFIL	278	
September	1981	ROTATE	154	
		ROTATE2	154	
		MEMXPD/SRC	174	
		CONVRT/SRC	186	
		HIDPICS	204	
		SORTER	220	
		SLPTSLPC/SRC	228	
		HRSRCE	244	
		ERRPRT	250	
		LIFETIME	252	
		DISKCHRGR	286	
		CARTOON	320	
		BASCAL	334	

continues

continued

Month	Year	Program Name	Page Number	
		LASTEST	136	(Pub. August 1981)
October	1981	MASTMIND	122	
		SYLLOG	132	
		VOICE	141	
		SPEAK	141	
		TYPE	141	
		TALKER	146	
		BSKBALL	184	
		HOMEINVT	220	
		SUNSET	272	
		FREEZER	274	
		TRSCPPM	288	
		YAHTZEE	302	
		BRIDGE	306	
		FIREFGHT	316	
		SPACE	334	
		JERICHO	292	(Pub. September 1981)
		TPRGEN/SRC	196	
		AUTOKEY/SRC	242	
		ASMPTCH/SRC	326	
		EDASFIX/SRC	344	
		COPYIT/SRC	370	
November	1981	TENANT	138	
		INVEST	144	
		HOUSE	160	
		GRAPHS	176	
		PRODTIMER	220	
		CHAIN	250	
		ALTER	274	
		CREDUN	321	
		SPRLIST	333	
		SECSRC	348	
		ENHANCE	384	
December	1981	LOGBOOK	100	
		STAMPS	108	
		PLANT1	116	
		PLANT2	116	
		COINS	134	
		CHESS	154	
		HEAT	220	
		GOLF	226	
		SENTRY	242	
		BLCKBK	278	
		DDENTRY	280	
		WRITER	290	
		CITY	310	
		ANINMAN	314	
		QUEUE	368	
		LOGGER	374	
		LUNANCY	386	
		ORGANIZE	398	
January	1982	PALLETS	98	
		SURVEY	102	
		RULE300	116	
		MODELT	140	
		ALPHGRPH	190	
		NEATLIST	196	
		SCRIPGRAFT	230	
		TAPESPTE	240	
		MARS	265	

Month	Year	Program Name	Page Number	
		VARMAP	304	
February	1982	GSETTR	70	
		PUNCTUAT	82	
		TESTMAKR	104	
		PLANETS	112	
		CAPCHEM	116	
		BOGGLE	166	
		GETTEAM	196	
		KINGSCAT	232	
		SALECAST	268	
		SNAPSHOT	272	
March	1982	THREEDEE	138	
		PRMSRCH	152	
		HUMANS	154	
		GRADES	184	
		SCRNPRNT	214	
		CENSOR	216	
		OTHLLO80	252	
		RGRAPHIC	294	
		BLTNBRD	314	
		FLPYFIXR	326	(Pub. December 1981)
April	1982	BASICK	86	
		BAM	167	
		RSL	188	
		ZUBRTNES	208	
		DISKRD3	214	
		COMMAND	244	
		MASDIR3	250	
		SHOPPING	270	
		RULER	302	
		CP80	306	
May	1982	MUSIC	142	
		NAMESONG	234	
		EXPENSE	238	
		RACES	252	
		HANDICAP	264	
		BANNER	282	
		SPCCHASE	292	
		GABBY	322	
		MONEYDOS	382	
June/July	1982	SMPLZAP1	232	
		SMPLZAP2	232	
		SMPLZAP3	232	
		SURVEY	248	
		QUESTION	248	
		LOADFILE	248	
		PPREDICT	272	
		DIRECTRY	288	
		AIRTRAVL	300	
		LEARNA	326	
		MERLIN	330	
		PROJCAL	348	
August	1982	HFLYNN46	66	
Games Issue		SPY	80	
		SPYFIELD	80	
		CUBE80	106	
		SLALOM	112	
		KALAH/SRC	132	
		LUCKY13	156	

continues

continued

Month	*Year*	*Program Name*	*Page Number*	
		ROUNDEND	156	
		STAR	156	
		DBLTRI	156	
		SUBCHOP2	216	
		SLOTMACH	222	
		CRAM	234	
		ACROSTIC	240	
		KUNGFU	248	
		MEMALPHA	254	
		TERMITES	274	
		PITTYPAT	270	
		THRUASTR	280	
		LOCO	286	
		RAMMER	296	
		CARDONI2	302	
September	1982	MOBYDICK	139	
		COLORART	168	
		TRAINART	168	
		CHARTART	168	
		BTFLYART	168	
		CSTLEART	168	
		SNPYART	168	
		SCRNPLAY	204	
		ORDRFORM	270	
		SIM	276	
		VEIL/SRC	286	
		HOTDOG	330	
		MTRMOUTH	330	
		PACHINKO	330	
		FUNHOUSE	330	
		HAMMING	342	
		MDOS1	364	
		MDOS2	364	
		SQUARE1	358	(Pub. August 1982)
		SQUARE2	358	(Pub. August 1982)
		SQUARE3	358	(Pub. August 1982)
		SQUARE4	358	(Pub. August 1982)
October	1982	QUIKCALC	114	
		BASICOMP	122	
		AUTOTRK2	203	
		MUSIC1	228	
		MUSIC2	228	
		SCRPTCH1/SRC	276	
		SCRPTCH2/SRC	276	
		CRYPTO	296	
		INVENTRY	342	
November	1982	MEDHIST	90	
		REGRESS	96	
		HEATSTRS	148	
		EINSTEIN	222	
		DRAWPOKR	246	
		BABYCASS	282	
		KWIKMAZE	318	
		PIXLPRNT	364	
		GENEOLOGY	398	
		ANCESTRY	398	
		FAMILY	398	
		TRON/SRC	406	
		QUIKSORT/SRC	414	
		BKINVTRY	436	
		KTAABA	450	

80 micro BOOKSHELF

HOBBY COMPUTERS ARE HERE! If you want to come up to speed on how computers work—hardware and software—this is an excellent book. It starts with fundamentals and explains the circuits and the basics of programming, along with a couple of TVT construction projects, ASCII, Baudot, etc. This book has the highest recommendations as a teaching aid. $2.97.* BK7322

THE NEW HOBBY COMPUTERS—This book takes up from where Hobby Computers Are Here! leaves off, with chapters on Large-Scale Integration, how to choose a microprocessor chip, an introduction to programming, low-cost I/O for a computer, computer arithmetic, checking memory boards . . . and much, much more! Don't miss this tremendous value! BK7340 Only $2.97.*

BOTH VOLUMES FOR $4.95!
ORDER HC002

NEW NEW NEW

INTRODUCTION TO WORD PROCESSING by Hal Glatzer. This book explains in plain language what a word processor can do, how to use one, how it improves productivity—especially in businesses that handle lots of words—and how to buy one wisely. No technical knowledge required, for all first-time users and those considering purchasing a word processor. BK1238 $12.95

A USER GUIDE TO THE UNIX SYSTEM by Jean Yates and Rebecca Thomas. Here at last is a clearly written book that allows you to use the Unix operating system easily, and at a fraction of the time it previously took. If you're using, evaluating or simply curious about this system, this is your book. BK1242 $15.99

WORDSTAR MADE EASY by Walter A. Ettlin. Now WordStar is as simple to learn as it is easy to use. This book teaches WordStar in 14 easy lessons, saving hours of hard work. It comes with a convenient pull-out Command Card. BK1239 $11.95

● **SOME OF THE BEST FROM KILOBAUD MICROCOMPUTING**—BK7311—A collection of the best articles that have appeared in Kilobaud MICROCOMPUTING. Included is material on the TRS-80 and PET systems, CP/M, the 8080/8085/Z-80 chips, the ASR-33 terminal. Data base management, word processing, text editors and file structures are covered too. Programming techniques and hard-core hardware construction projects for modems, high speed cassette interfaces and TVTs are also included in this large format, 200 plus page edition $10.95 *

DON'T (or How to Care for Your Computer)—by Rodnay Zaks. In plain language, with numerous illustrations, this book tells all the do's and don't's of the care, preservation and correct operation of the small computer system. Specific chapters cover each piece of hardware and software, as well as safety and security precautions and help for problem situations. Have your computer work right the first time and keep it working. No technical background required. For all computer users. BK1237 $11.95.

YOUR FIRST COMPUTER—by Rodnay Zaks. Whether you are using a computer, thinking about using one or considering purchasing one, this book is indispensable. It explains what a computer system is, what it can do, how it works and how to select various components and peripheral units. It is written in everyday language and contains invaluable information for the novice and the experienced programmer. (The first edition of this book was published under the title "An Introduction to Personal and Business Computing".) BK1191 $8.95*

MICROPROCESSOR INTERFACING TECHNIQUES—by Austin Lesea & Rodnay Zaks—will teach you how to interconnect a complete system and interface it to all the usual peripherals. It covers hardware and software skills and techniques, including the use and design of model buses such as the IEEE 488 or S-100. BK1037 $17.95.*

AN INTRODUCTION TO MICROCOMPUTERS, VOL. 0—The Beginner's Book—Written for readers who know nothing about computers—for those who have an interest in how to use computers—and for everyone else who must live with computers and should know a little about them. The first in a series of 4 volumes, this book will explain how computers work and what they can do. Computers have become an integral part of life and society. During any given day you are affected by computers, so start learning more about them with Volume 0. BK1130 $12.50.*

VOL. I—NEW 3rd Edition! Dedicated to the basic concepts of microcomputers and hardware theory. The purpose of Volume 1 is to give you a thorough understanding of what microcomputers are. From basic concepts (which are covered in detail), Volume 1 builds the necessary components of a microcomputer system. This book highlights the difference between minicomputers and microcomputers. BK1030 $15.95

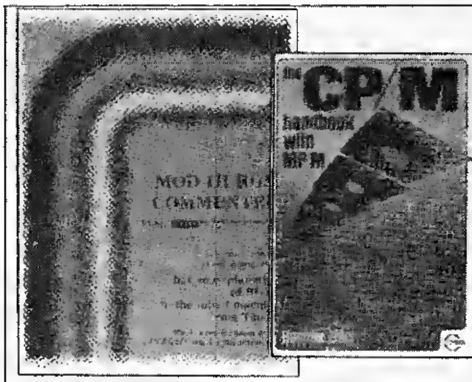
KILOBAUD KLASSROOM—by George Young and Peter Stark. Learning electronics theory without practice isn't easy. And it's no fun to build an electronics project that you can't use. *Kilobaud Klassroom*, the popular series first published in *Kilobaud Microcomputing*, combines theory with practice. This is a *practical* course in digital electronics. It starts out with very simple electronics projects and by the end of the course you'll construct your own working microcomputer!

Authors Young and Stark are experienced teachers, and their approach is simple and direct. Whether you're learning at home or in the classroom, this book provides you with a solid background in electronics—and you'll own a computer that you built yourself! BK7386 $14.95

TOOLS & TECHNIQUES FOR ELECTRONICS—by A. A. Wicks is an easy-to-understand book written for the beginning kit-builder as well as the experienced hobbyist. It has numerous pictures and descriptions of the safe and correct ways to use basic and specialized tools for electronic projects, as well as specialized metal-working tools and the chemical aids which are used in repair shops. $4.95.* BK7348

● **UNDERSTANDING AND PROGRAMMING MICROCOMPUTERS**—BK7382—A valuable addition to your computing library This two-part text includes the best articles that have appeared in 73 and Kilobaud Microcomputing magazines on the hardware and software aspects of microcomputing Well-known authors and well-structured text helps the reader get involved. $10.95*

Special Interests



DOS RANDOM ACCESS & BASIC FILE HANDLING—By H J. Muller. This book was written for the nonprogrammer. It is ideal for the businessman or professional who needs to solve and write special programs for in-house business problems, or the hobbyist who wants to go beyond the cassette recorder and into disk storage and file manipulation. It is written as a self-instruction tutorial and will provide anyone with some Level II experience with the ability to write special programs for inventories, mailing list, work scheduling, record keeping, research project data manipulation, etc. 150 pages. BK1236 $29.50

TRS-80 DISK AND OTHER MYSTERIES—by Harvard C. Pennington. This is the definitive work on the TRS-80 disk system. It is full of detailed "How to use," information with examples, samples and in-depth explanations suitable for beginners and professionals alike. The recovery of one lost file is worth the price alone. BK1181 $22.50.*

MICROSOFT BASIC DECODED AND OTHER MYSTERIES—by James Farvour. From the company that brought you *TRS-80 DISK AND OTHER MYSTERIES*. Contains more than 6500 lines of comments for the disassembled Level II ROMs and six additional chapters describing every BASIC subroutine, with assembly language routines showing how to use them. Flowcharts for all major routines give the reader a real insight into how the interpreter works. BK1186 $29.50.*

MOD III ROM COMMENTED—Soft-Sector marketing, 1981. This book is not an Instruction course on machine language, but rather an information source that you can use time and time again for writing your own program or patching old Mod I machine language programs. It contains an explanation of ROMs in the latest machine from Tandy, with most every location of the 14K ROMs listed, with comments. BK1235 $22.50.

THE CUSTOM TRS-80 AND OTHER MYSTERIES—by Dennis Kitsz. More than 300 pages of TRS-80 customizing information. With this book you'll be able to explore your computer like never before. Want to turn an 8 track into a mass storage unit? Individual reverse characters? Replace the BASIC ROMs? Make Music? High speed, reverse video, Level I and Level II? Fix it if it breaks down? All this and much, much more. Even if you have never used a soldering iron or read a circuit diagram, this book will teach you how! This is the definitive guide to customizing your 80! BK1218 $29.95.*

BASIC FASTER AND BETTER AND OTHER MYSTERIES—by Lewis Rosenfelder. You don't have to learn assembly language to make your programs run fast. With the dozens of programming tricks and techniques in this book you can sort at high speed, swap screens in the twinkling of an eye, write INKEY routines that people think are in assembly language and add your own commands to BASIC. Find out how to write elegant code that makes your BASIC really hum, and explore the power of USR calls. BK1221 $29.95.*

THE CP/M HANDBOOK (with MP/M)—by Rodnay Zaks. A complete guide and reference handbook for CP/M—the industry standard in operating systems. Step-by-step instruction for everything from turning on the system and inserting the diskette to correct user discipline and remedial action for problem situations. This also includes a complete discussion of all versions of CP/M up to and including 2.2, MP/M and CDOS. BK1187 $14.95.*

PRICES SUBJECT TO CHANGE WITHOUT NOTICE

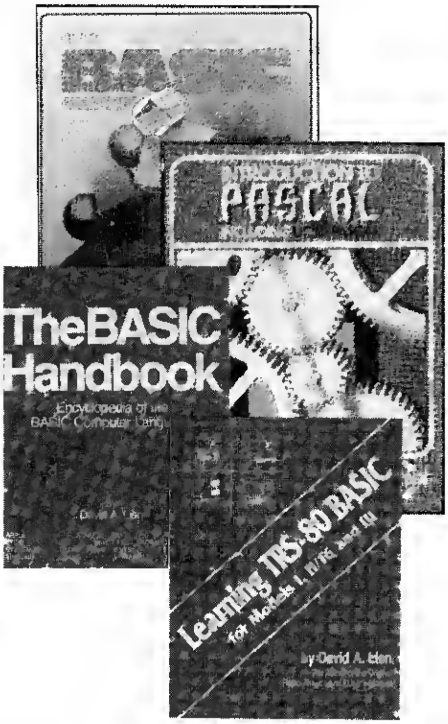
No C.O.D. orders accepted. All orders add $1.50 for first book, $1.00 each additional book, $10.00 per book foreign airmail. Please allow 4-6 weeks for delivery. Questions regarding your order? Please write to Customer Service at the following address. Use the order card in this magazine or itemize your order on a separate piece of paper and mail to *80 Micro* Book Department•Peterborough NH 03458. Be sure to include check or detailed credit card information.

For Toll Free Ordering Call 1-800-258-5473

✓See List of Advertisers on Page 563

80 micro BOOKSHELF

Basic & Pascal



INTRODUCTION TO TRS-80 LEVEL II BASIC AND COMPUTER PROGRAMMING—by Michael P. Zabinski. Written by an experienced educator, this is the book for those beginners who want to learn about computers without having to become an expert. It has practical programs, useful line-by-line comments, excellent flowcharts accompanied by line numbers and over 200 exercises which help the reader assess progress, reinforce comprehension, and provide valuable practical experience. BK1219 $10.95.*

50 BASIC EXERCISES—by J. P. Lamoitier. This book is structured around the idea that the best way to learn a language is through actual practice. It contains 50 completely explained exercises: statement and analysis of the problem, flowcharts, programs and actual runs. Program subjects include mathematics, business, games, and operations research, and are presented in varying levels of difficulty. This format enables anyone to learn BASIC rapidly, checking their progress at each step. BK1192 $12.95*.

NEW REVISED EDITION

PROGRAMMING IN PASCAL—by Peter Grogono. The computer programming language PASCAL was the first language to embody in a coherent way the concepts of structured programming, which has been defined by Edsger Dijkstra and C.A.R. Hoare. As such, it is a landmark in the development of programming languages. PASCAL was developed by Niklaus Wirth in Zurich; it is derived from the language ALGOL 60 but is more powerful and easier to use. PASCAL is now widely accepted as a useful language that can be efficiently implemented, and as an excellent teaching tool. It does not assume knowledge of any other programming language and therefore suitable for an introductory course. BK1140 $12.95.*

ADVANCED BASIC—Applications including strings and files, coordinate geometry, area, sequences and series, simulation and graphing and games. BK1000 $11.50

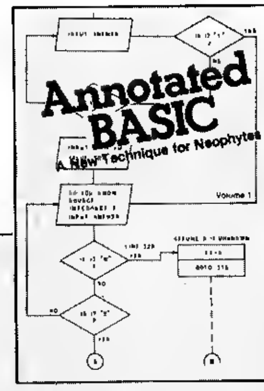
INTRODUCTION TO PASCAL—By Rodnay Zaks. A step-by-step introduction for anyone wanting to learn the language quickly and completely. Each concept is explained simply and in a logical order. All features of the language are presented in a clear, easy-to-understand format with exercises to test the reader at the end of each chapter. It describes both standard PASCAL and UCSD PASCAL, the most widely used dialect for small computers. No computer or programming experience is necessary. BK1189 $15.95

THE BASIC HANDBOOK—SECOND EDITION—by David Lien. This book is unique. It is a virtual ENCYCLOPEDIA of BASIC. While not favoring one computer over another, it explains over 250 BASIC words, how to use them and alternate strategies. If a computer does not possess the capabilities of a needed or specified word, there are often ways to accomplish the same function by using another word or combination of words. That's where the HANDBOOK comes in. It helps you get the most from your computer, be it a "bottom-of-the-line" micro or an oversized monster. BK1174 $19.95.*

LEARNING TRS-80 BASIC–by David A. Lien. Dr. Lien, who is the author of THE BASIC HANDBOOK and the original Radio Shack LEVEL I USER'S MANUAL, has compiled a tutorial which includes portions of the original USER'S MANUAL, and most of LEARNING LEVEL II along with extensive additions. It will completely cover the TRS-80 Models I, II, III, and 16 (sorry, not the color or pocket computers). It is, of course, written in the easy learning style which readers of Dr. Lien's books have come to enjoy. BK1175 $19.95.

BASIC BASIC (2ND EDITION)—by James S. Coan. This is a textbook which incorporates the learning of computer programming using the BASIC language with the teaching of mathematics. Over 100 sample programs illustrate the techniques of the BASIC language and every section is followed by practical problems. This second edition covers character string handling and the use of data files. BK 1026 $11.50.*

NEW NEW NEW



ANNOTATED BASIC—A NEW TECHNIOUE FOR NEOPHYTES.—BASIC programming was supposed to be simple—a beginner's programming language which was so near to English that is could be easily understood. But, in recent years, BASIC has become much more powerful and therefore much more difficult to read and understand. BASIC simply isn't basic anymore.

Annotated BASIC explains the complexities of modern BASIC. It includes complete TRS-80* Level II BASIC programs that you can use. Each program is annotated to explain in step-by-step fashion the workings of the program. Programs are flowcharted to assist you in following the operational sequence. And—each chapter includes a description of the new concepts which have been introduced.

Annotated BASIC deals with the hows and whys of TRS-80 BASIC programming. *How* is a program put together? *Why* is it written that way? By observing the programs and following the annotation, you can develop new techniques to use in your own programs—or modify commercial programs for your specific use.

Annotated BASIC Volume 1 BK7384 $10.95
Annotated BASIC Volume 2 BK7385 $10.95
Order Both Volumes and Save! BK738402 $18.95

Moneymaking

HOW TO MAKE MONEY WITH COMPUTERS—In 10 information-packed chapters, Jerry Felsen describes more than 30 computer-related, money-making, high profit, low capital investment opportunities. BK1003 $15.00.*

HOW TO SELL ANYTHING TO ANYBODY—According to *The Guinness Book of World Records*, the author, Joe Girard, is "the world's greatest salesman." This book reveals how he made a fortune—and how you can, too. BK7306 $2.25.*

THE INCREDIBLE SECRET MONEY MACHINE—by Don Lancaster. A different kind of "cookbook" from Don Lancaster. Want to slash taxes? Get free vacations? Win at investments? Make money from something that you *like* to do? You'll find this book essential to give you the key insider details of what is really involved in starting up your own money machine. BK1178 $5.95.*

Games

CAPTAIN 80 BOOK OF BASIC ADVENTURES—by Robert Liddil. This book contains 18 of the most popular Adventure programs available today; all in one easy to read book with listings ready to be keyed in to your computer. This unique book also contains an ADVENTURE GENERATOR program, not available from any other source. This generator program will actually write another BASIC ADVENTURE PROGRAM! Although specifically written for the TRS-80 Model I & III, these programs are adaptable to other computers using Microsoft BASIC. BK 1240 $19.95.

40 COMPUTER GAMES FROM *KILOBAUD MICROCOMPUTING*—Forty games in all in nine different categories. Games for large and small systems, and even a section on calculator games. Many versions of BASIC used and a wide variety of systems represented. A must for the serious computer gamesman. BK7381 $7.95.*

BASIC COMPUTER GAMES—Okay, so once you get your computer and are running in BASIC, then what? Then you need some programs in BASIC, that's what. This book has 101 games for you from very simple to real buggers. You get the games, a description of the games, the listing to put in your computer and a sample run to show you how they work. Fun. Any one game will be worth more than the price of the book for the fun you and your family will have with it. BK1074 $7.50.*

MORE BASIC COMPUTER GAMES—Edited by David H. Ahl. More fun in BASIC! 84 new games from the people who brought you *BASIC Computer Games*. Includes such favorites as Minotaur (battle the mythical beast) and Eliza (unload your troubles on the doctor at bargain rates). Complete with game description, listing and sample run. BK1182 $7.50.*

WHAT TO DO AFTER YOU HIT RETURN—PCC's first book of computer games. . .48 different computer games you can play in BASIC. . . programs, descriptions, and many illustrations. Lunar Landing, Hammurabi, King, Civel 2, Qubic 5, Taxman, Star Trek, Crash, Market, etc. BK1071 $16.50*.

Business

BUSINESS SYSTEM BUYER'S GUIDE—By Adam Osborne with Steven Cook. When you enter the market place of small business computers, you face a bewildering array of products, prices, features and fables. This guide cuts through the jargon and unravels the task of buying the right computer system. This book provides solid information on how to determine your needs, how to choose software and hardware for all business applications, what to expect from vendors, what to avoid, and what questions you must ask. It also provides a wealth of detailed information on products, manufacturers, retailers and the whole microcomputer market. BK1229 $9.95

SO YOU ARE THINKING ABOUT A SMALL BUSINESS COMPUTER—by Richard G. Canning and Nancy C. Leeper. For a well-organized manual on the process of selecting the right computer system for your small business, this text can't be excelled. Designed to introduce the novice in data and word processing to the real benefits of computerization, the book is filled with money- and time-saving tips, photos of equipment, lists of suppliers, prices, explanations of computer terminology, and helpful references to additional sources of information. Everyone contemplating a first computer installation should have this book. BK1222 $14.00*

SOME COMMON BASIC PROGRAMS— Published by Adam Osborne & Associates, Inc. Perfect for non-technical computerists requiring ready-to-use programs. Business programs, plus miscellaneous programs. Invaluable for the user who is not an experienced programmer. All will operate in the stand-alone mode. BK1053 $14.99 paperback.

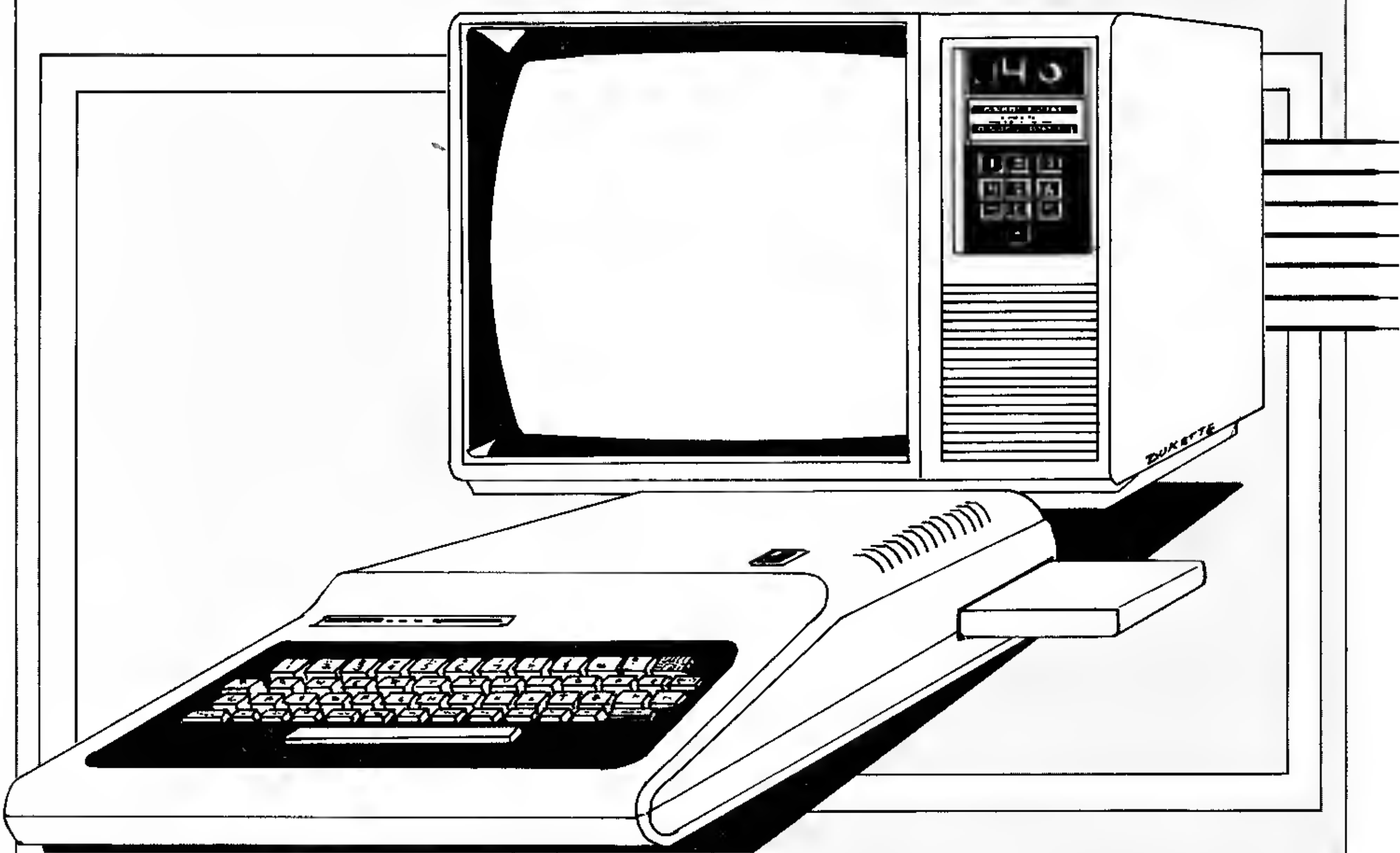
THEORY Z—How American Business Can Meet the Japanese Challenge—by William Ouchi. Why are the Japanese catching up and surpassing American industrial productivity? What allows Japanese industrialists to offer guaranteed lifetime employment to their workforce? This book will help you understand the Theory Z managerial philosophy and its implications for the American corporate future. Examples are given of the American industrial giants already operating under Z-style management, and the impact of this style on the quality of their executives and workers is explored. A must for the alert businessman, large or small. BK1226 $12.95*

No C.O.D. orders accepted. All orders add $1.50 for first book, $1.00 each additional book, $10.00 per book foreign airmail. Please allow 4-6 weeks for delivery. Questions regarding your order? Please write to Customer Service at the following address. Use the order card in this magazine or itemize your order on a separate piece of paper and mail to *80 Micro* Book Department•Peterborough NH 03458. Be sure to include check or detailed credit card information.

For Toll Free Ordering Call 1-800-258-5473

Part III.

COLOR COMPUTER



Teach Your CC New Words

by Allen Curtis

Add commands, statements, and functions to Color Basic and gain more computing power.

The size and versatility of the Color Computer's Basic vocabulary is more than adequate for the majority of programs. However, there are occasions when it would be convenient to use statements, commands,or functions not available in the Color Computer's vocabulary.

Microsoft, writers of the Color Computer's Basic interpreter, have developed a systematic procedure for adding Extended Color Basic key words to the Color Basic vocabulary, and in turn adding Disk Basic key words to the Extended Color Basic vocabulary. I will describe this procedure. I will also demonstrate a more economical way of drawing and animating graphics.

ROM Tables

A machine-language subroutine corresponds to each Basic word in the Color Computer's vocabulary in the associated ROM. In the Color Basic ROM there are two tables of Color Basic key words and two associated tables of addresses. The addresses are the entry addresses to the subroutines that execute the statements, commands, or functions associated with the Basic key words. The first Basic key-word table is

> ### *The Key Box*
> **Color Computer**
> **16K RAM**
> **Extended Color Basic**

Color Basic Word	Subroutine Entry	Token
FOR	AD47	80
GO	AE86	81
REM	AEE3	82
'	AEE3	3A83
ELSE	AEE3	3A84
IF	AF14	85
DATA	AEE0	86
PRINT	B8F7	87
ON	AF42	88
INPUT	AFF5	89
END	AE02	8A
NEXT	B0F8	8B
DIM	B34E	8C
READ	B046	8D
RUN	AE75	8E
RESTORE	ADE4	8F
RETURN	AEC0	90
STOP	AE09	91
POKE	B757	92
CONT	AE30	93
LIST	B764	94
CLEAR	AE41	95
NEW	AD17	96
CLOAD	A498	97
CSAVE	A44C	98
OPEN	A5F6	99
CLOSE	A416	9A
LLIST	B75E	9B
SET	A880	9C
RESET	A8B1	9D
CLS	A910	9E
MOTOR	A7BD	9F
SOUND	A94B	A0
AUDIO	A990	A1
EXEC	A53E	A2
SKIPF	A5EC	A3
TAB(		A4
TO		A5
SUB		A6
THEN		A7
NOT		A8
STEP		A9
OFF		AA
+		AB
−		AC
*		AD
/		AE
		AF
AND		B0
OR		B1
>		B2
=		B3
<		B4

Table 1

Turn your color computer on to the power of FLEX and OS-9
NOW FROM THE WORLDS LARGEST SUPPLIER OF SOFTWARE FOR FLEX AND OS-9 COMES FHL COLOR FLEX AND COLOR OS-9, JUST LOOK AT THESE FEATURES:
OS-9 ONLY $200
OS-9 WITH BASIC09 $299
• Unix*-based, multitasking, modular, and versatile
• Over 40 utility commands
• Friendly "Shell" command interpreter
• Tree structured multilevel file directories
• Full time sharing support with log in and file security
• Fast, secure, random and sequential access files
• Comprehensive English language error messages
• Compact real-time multitasking executive
• BASIC09 is the fastest and most comprehensive full Basic language available for the 6809. It combines standard Basic with the best features of Pascal. It's a unique interactive compiler that combines compiler speed, interpreter friendliness, and superlative debugging facilities.
FLEX NOW ONLY $99
• HI-RES screen formats
• 16 x 32 and 24 x 51, upper and lower case characters
• 24 x 64 and 32 x 64 upper case
• Full ASCII keyboards
• Easy start-up—just type RUN "FLEX"
• Online assistance just type HELP
• Optionally use a standard terminal and printer
• Advance disk I/O and terminal capabilities
• NO additional hardware required
• We have supported FLEX with more software than anyone else in the world for more than 2 years!
SPECIAL
1. DBASIC, RS Disk Basic under FLEX with a utility to copy RS to FLEX disk $35.
2. EDASM line and screen editor and macro assembler, both more powerful than TSC's, and at the same cost, only $100.
3. UTILITIES, a set of 12 utilities especially designed for FHL Color FLEX $50.
FLEX
For the TRS-80 Color Computer
FRANK HOGG LABORATORY, INC.
Get on our mailing list, call or send for our complete catalog of over 100 products for FLEX and OS-9. We're doing exciting things with your color computer!
✓ 252
FRANK HOGG LABORATORY
THE REGENCY TOWER • 770 JAMES ST. • SYRACUSE, NY 13203 TELEPHONE (315) 474-7856
*Unix is a trademark of Bell Labs. *OS-9 & BASIC09 are trademarks of Microware & Motorola Inc. *FLEX is a trademark of Technical Systems Consultants, Inc.

located at addresses AA66–AB19, where the addresses are in hexadecimal notation.

The table of subroutine entry addresses associated with the key-word table is located at addresses AB67–ABAE (Table 1). In Table 1 the Token column is not part of either ROM table, but I included it for the sake of completeness.

When you key in a Basic program, the Basic key words you type are not stored in that form in memory. To conserve memory space, the Basic interpreter encodes these key words into tokens. The tokens for the Basic key words in Table 1 range from hexadecimal 80–B4. The Basic interpreter uses the ROM tables in Table 1 to translate each key word therein to its corresponding token.

The ROM table of subroutine addresses does not include addresses for the final 17 words of Table 1. These key words are suffixes and operators whose subroutines are parts of other ROM subroutines. Assembly-language programmers wanting to incorporate some of the subroutines into their programs will find the information in Tables 1–6 helpful.

The other word table in the Color Basic ROM is devoted to key words for functions. This table starts at address AB1A and ends at AB66. Its associated subroutine entry-address table is locat-

Color Basic Word	Subroutine Entry	Token
SGN	BC7A	FF80
INT	BCEE	FF81
ABS	BC93	FF82
USR	0112	FF83
RND	BF1F	FF84
SIN	BF78	FF85
PEEK	B750	FF86
LEN	B681	FF87
STR$	B4FD	FF88
VAL	B716	FF89
ASC	B6A0	FF8A
CHR$	B68C	FF8B
EOF	A5CE	FF8C
JOYSTK	A9C6	FF8D
LEFT$	B6AB	FF8E
RIGHT$	B6C8	FF8F
MID$	B6CF	FF90
POINT	A8F5	FF91
INKEY$	A564	FF92
MEM	B4EE	FF93

Table 2

Extended Color Basic Word	Subroutine Entry	Token
DEL	8970	B5
EDIT	8533	B6
TRON	86A7	B7
TROFF	86A8	B8
DEF	8871	B9
LET	AF89	BA
LINE	93BB	BB
PCLS	9532	BC
PSET	9361	BD
PRESET	9365	BE
SCREEN	9670	BF
PCLEAR	968B	C0
COLOR	9546	C1
CIRCLE	9E9D	C2
PAINT	98EC	C3
GET	9755	C4
PUT	9758	C5
DRAW	9CB6	C6
PCOPY	9723	C7
PMODE	9621	C8
PLAY	9A22	C9
DLOAD	8C18	CA
RENUM	8A09	CB
FN		CC
USING		CD

Table 3

Extended Color Basic Word	Subroutine Entry	Token
ATN	8EB0	FF94
COS	8378	FF95
TAN	8381	FF96
EXP	84F2	FF97
FIX	8524	FF98
LOG	8446	FF99
POS	86AC	FF9A
SQR	8480	FF9B
HEX$	8BDD	FF9C
VARPTR	86BE	FF9D
INSTR	877E	FF9E
TIMER	8968	FF9F
PPOINT	9339	FFA0
STRING$	874E	FFA1

Table 4

Disk Basic Word	Subroutine Entry	Token
DIR	CBCF	CE
DRIVE	CDE9	CF
FIELD	CFE0	D0
FILES	D080	D1
KILL	C6C2	D2
LOAD	C99A	D3
LSET	D026	D4
MERGE	C98B	D5
RENAME	CF3F	D6
RSET	D025	D7
SAVE	C932	D8
WRITE	CF8A	D9
VERIFY	D65B	DA
UNLOAD	D146	DB
DSKINI	D4AB	DC
BACKUP	D175	DD
COPY	D2CC	DE
DSKI$	D3FF	DF
DSKO$	D474	E0

Table 5

Disk Basic Word	Subroutine Entry	Token
CVN	CD1A	FFA2
FREE	CDC0	FFA3
LOC	CD36	FFA4
LOF	CD5B	FFA5
MKN$	CD28	FFA6
AS	B277	FFA7

Table 6

ONLY $69.95

The Magnificent WORD PROCESSING SYSTEM

For the TRS-80 Model I and III

- Written in fast Z80 machine language
- Single key control of all editing functions for ease of use.
- Dynamic display of word count, line count, and free memory count.
- Superscripts, subscripts, underlined, bolded, expanded and condensed type styles - combine and intermix within a line
- Automatically justifies and word-wraps on the screen as you type
- Search, Replace, and Global Search and Replace
- Odd and even page user-definable headers, footers, and page number lines
- User-definable linespacing, sheet size, top, left, and bottom margins
- Move blocks of text from disk, to disk, and within the text
- Examine disk directory on any disk and kill files while editing
- Full screen editing of EDTASM and BASIC text files.
- Automatic renumbering for EDTASM and BASIC files
- Print contents of screen function
- Print previewing formats text, inserts headers, numbers pages, etc. on the screen without printing it on paper
- Page by page pausing capability for sheet fed printers
- Supports both parallel and serial printers
- Printer control code access
- Supports proportional space justifying on Centronics 737, 739, Line Printer IV, Daisy Wheel II, Diablo, Qume, Starwriter, Spinwriter, C. Itoh Prowriter, NEC PC-8023A-C, TEC 8500R
- Also supports special functions of Microline, MX-80, MX-100, and Graftrax Plus
- Any character or symbol your printer can print is accessable with the Special Character feature.
- Works with NEWDOS, NEWDOS80, TRSDOS, LDOS, and DOSPLUS - Single or Double Density

GUARANTEE

Many word processing systems claim theirs are the best, but few would dare to guarantee them. Not us! We are confident that **ZORLOF** is the most useful word processing system on the market for under $200.00. If you don't agree, return it within 30 days for a full refund.

CALL (305) 259-9397 ✓141

Add $2.00 shipping & handling. Florida residents add 5% sales tax. Checks require 3 weeks to clear banks.

ANITEK — MasterCard — VISA

ANITEK SOFTWARE PRODUCTS □ P.O. BOX 1136 □ MELBOURNE, FL 32935 □ (305)259-9397

TRS-80™ Model 1 & 3 Software

SYSTEM DIAGNOSTIC $99.95

Tests every component of your TRS-80 Model 1 or 3: ROM, RAM, Video Display, Keyboard, Cassette Recorder, Line Printer, Disk Drives (single or double density, 35, 40, or 80-track drives), RS-232-C Interface.

SMART TERMINAL $74.95
Model II CP/M Version $79.95

The intelligent telecommunication program. Automatic transmission and reception of files. True BREAK key. Cassette and disk files compatible with both SCRIPSIT™ and Electric Pencil™. Same program supports both cassette and disk systems.

HOME BUDGET $49.95
Cassette Version $29.95

Analyzes your income and expenditures, and prints monthly and year-to-date summaries showing income tax deductions. Complete instructions on customizing to your own budget.

SMALL BUSINESS ACCOUNTING $59.95
Cassette Version $29.95

Based on Dome Bookkeeping Record #612, keeps track of income, expenditures, and payroll for a small business. Income and expenses can be entered on a daily, weekly, or monthly basis. Program computes monthly, through last month, and year-to-date summaries.

MAILING LIST $69.95

Maintains mailing lists of up to 1326 names on a single formatted diskette. Add, change, delete, find names, sort by information in any field, print up to 4-line labels of 32 characters per line on 1, 2, 3, or 4 column labels.

Documentation to any program can be purchased separately for $10.00, deductible from later software order.

Specify TRS-80 Model 1 or 3.
Add $3.00 for postage and handling
New York State residents add 4 ¼% sales tax.

HOWE SOFTWARE

14 Lexington Road, New City, NY 10956

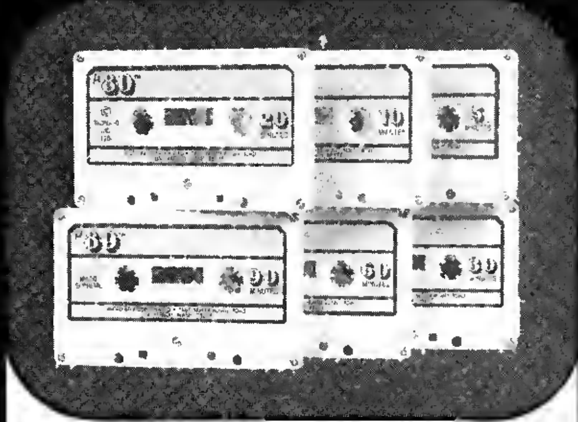
(914) 634-1821

Visa and MasterCard accepted.

TRS-80 is a registered trademark of Tandy Corp. ✓175

PREMIUM CASSETTES

- COMPUTER GRADE
- 100% ERROR-FREE
- FULLY GUARANTEED



The Micro-Trac™ Generation

- High Frequency Response
- Wide Dynamic Range
- *Used by Software Firms & Computer Hobbyists*
- Choice of School Districts Nationwide

LENGTH	12-PAK	24-PAK
C-05	$ .69	$ .59
C-10	$ .79	$ .69
C-20	$ .99	$ .89
C-30	$1.19	$1.09
C-60	$1.49	$1.39
C-90	$1.79	$1.69
*CASES	$ .21	$.20

*Cases recommended to protect sensitive cassettes.

—UPS SHIPPING—
(No P.O. Boxes please)
$2.00 (12-pak) $3.00 (24-pak)
—Canadian shipping multiply by 2—

#1 COMPUTER TAPE IN THE USA!

—write for volume prices—

TOLL-FREE
(orders only)
1-800-528-6050
ext. 3005
In Arizona State
1-800-352-0458
ext. 3005

VISA — master charge THE INTERBANK CARD

MICRO-80™ INC.

E. 2665 Busby Road
Oak Harbor, WA 98277
—Distributors Wanted!—

✓132

✓See List of Advertisers on Page 563

ed at addresses AA29–AA50. These two ROM tables combine to form Table 2. The tokens for the function key words require two bytes each.

In the Extended Color Basic and Disk Basic ROMs, there are tables analogous to those in Tables 1 and 2. Their locations are at addresses 8183–81EF, 81F0–821D, 821E–8256, 8257–8272, C17F–C1DA, C1DB–C200, C201–C213, and C214–C21F. The substance of these ROM tables is in Tables 3, 4, 5, and 6.

Auxiliary RAM Control Bytes

In the auxiliary RAM (addresses 0000–03FF) of the Extended Color Basic ROM there are bytes used to control the Basic interpreter's action in determining from the current token which subroutine entry to make. These control bytes are located at addresses 0120–013D. At 0120 is the number of Color Basic key words in the ROM table that is part of Table 1. The two bytes at 0121 and 0122 point to this ROM table of Color Basic key words. The succeeding two bytes point to the associated ROM table of subroutine entry addresses.

The succeeding five addresses, 0125–0129 serve similar purposes for the ROM tables associated with Table 2. The next five bytes diverge somewhat from the established pattern. The byte at 012A contains the number of key words in the Extended Color Basic ROM shown by Table 3.

The bytes at 012B and 012C point to the ROM table of Extended Color Basic key words. The divergence occurs at 012D and 012E. These bytes point to a subroutine that helps determine the subroutine entry address for tokens B5–CB. The subroutine also takes care of tokens FF90 and FF9F.

The bytes at addresses 012F–0133 have a purpose similar to the previous five bytes. The final two bytes at 0132 and 0133 point to a subroutine that helps determine the subroutine entry addresses for tokens FF94–FFA1, with the exception of FF90 and FF9F.

The two subroutines that help in determining subroutine entry addresses also provide exits to addresses stored at 0137, 0138 and 013C, 013D. When the Color Computer does not have a Radio Shack Disk System, these latter addresses lead to a subroutine that sends the SN error message to the screen. When the machine does have a Disk System, the 10 bytes in locations 0134–013D have properties analogous to the preceding 10 bytes.

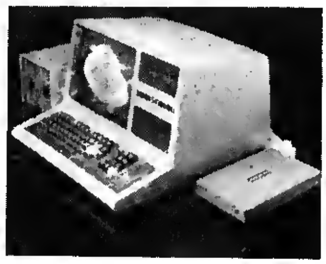
HPOKE, EXG, and EOR

The following two statements are extremely valuable additions to Extended Color Basic.

- A POKE-like statement that transfers hexadecimal bytes to successive RAM addresses: For example, HPOKE &H2900,CE,3E,9C,8E,C1,7F,C6,5C,BD,A5,9A sends the bytes CE, 3E,... 9A to addresses 2900–290A. HPOKE provides a convenient means of typing in machine-language programs. It also greatly simplifies the use of elegant string-packing techniques. HPOKE eliminates the need for the slow Read, Data, POKE loops used in string packing. Ironically, string packing is used in the program that adds HPOKE to the Extended Color Basic vocabulary. HPOKE is also the key to a new high-resolution graphics technique.

- A single statement that quickly does the equivalent of:

```
FORI=0TO3
B=PEEK(VARPTR(A$(X))+I)
POKEVARPTR(A$(X))+I,
```

PEEK(VARPTR(A$(Y)) + I)
POKEVARPTR(A$(Y)) + I,B
NEXT

without using variables I and B. This statement, called EXG, quickly exchanges two strings. Suppose A$ = "FIRST" and B$ = "SECOND". After the execution of EXG(A$,B$), then A$ = "SECOND" and B$ = "FIRST". The loop

FORI = A TON
EXG(A$(I),A$(I + I))
NEXT

can be invaluable in rapid string sorts.

A useful tool for specialists is the EOR function. C = EOR(A,B) executes twice as fast as its equivalent C = (NOTA AND B)OR(A AND NOTB).

Adding to Basic's Vocabulary

The following are the steps I use to add to the Basic vocabulary. You can use this general procedure to add words of your own.

First, write subroutines for HPOKE, EXG, and EOR and assign them to RAM locations. Choose high RAM addresses that you can protect using the Clear statement. For a 16K RAM, use

```
20 REM ** THERE ARE 210 SPACES IN STRING OF LINE 30. **
30 A$="


                                          "
40 X=VARPTR(A$)+2:X=256*PEEK(X)+PEEK(X+1)
50 FORI=X TOX+209
60 READJ
70 POKEI,J
80 NEXT
100 DATA2,&H3F,&H57,&H3F,&H68,1,&H3F,&H63,&H3F,&H77
110 DATA&H34,&H60,&H30,&H8C,&HF1,&HCE,1,&H34,&HC6,&HA,&HBD,&HA5,
&H9A,&H6F,&HC4,&H7F,1,&H43
120 DATA&HCE,&H3F,&H57,&H30,&H8C,7,&HC6,&HB8,&HBD,&HA5,&H9A,&H35
,&HE0
130 DATA&H48,&H50,&H4F,&H4B,&HC5,&H45,&H58,&HC7,&H3F,&H85,&H3F,&
HBC
140 DATA&H45,&H4F,&HD2,&H3F,&HE5
150 DATA&H81,&HD0,&H25,3,&H7E,&HB2,&H77,&H8E,&H3F,&H5F,&H80,&HCE
,&H7E,&HAD,&HD4
160 DATA&HC1,&H44,&H23,2,&H20,&HEF,&HC0,&H44,&H8E,&H3F,&H66,&H7E
,&HB2,&HCE
170 DATA&HBD,&HB7,&H3D,&H9D,&HA5,&H81,&H2C,&H26,&H2D
180 DATA&H8D,&H10,&H48,&H48,&H48,&H48,&H34,2,&H8D,8,&HAB,&HE0
190 DATA&HA7,&H80,&H9D,&H9F,&H20,&HE8,&H9D,&H9F,&H81,&H30,&H25,&
HC
200 DATA&H81,&H3A,&H25,&HF,&H81,&H41,&H25,4,&H81,&H47,&H25,5
210 DATA&H32,&H62,&H7E,&HB2,&H77,&H80,7,&H80,&H30,&H39
220 DATA&HBD,&HB2,&H6A,&HBD,&HB3,&H57,&H34,&H10
230 DATA&HBD,&HB2,&H6D,&HBD,&HB3,&H57,&H35,&H40,&HA6,&HC4
240 DATA&HE6,&H84,&HA7,&H81,&HE7,&HC1,&H8D,6,&H8D,4,&HBD,&HB2,&H
67
250 DATA&H39,&HA6,&HC4,&HE6,&H84,&HA7,&H80,&HE7,&HC0,&H39
260 DATA&HBD,&HB2,&H6A,&HBD,&HB7,&H3D,&H34,&H10,&HBD,&HB2,&H6D,&
HBD,&HB7,&H3D
270 DATA&HBD,&HB2,&H67,&H1F,&H10,&HA8,&HE0,&HE8,&HE0,&HBD,&HB4,&
HF4,&H39
```

Program Listing 1

the following starting and ending addresses for HPOKE, EXG, and EOR respectively: 3F85 and 3FBB, 3FBC and 3FE4, and 3FEF and 3FFF.

Second, construct a key-word table for HPOKE and EXG. The table consists of five hexadecimal bytes for HPOKE and three hexadecimal bytes for EXG. The first four bytes are the ASCII codes for the letters H, P, O, and K. The fifth byte is the ASCII code for E incremented by hexadecimal 80. The sixth and seventh bytes are the ASCII codes for E and X. The final byte is 80 plus the ASCII code for G.

The final letter of every Basic key word in the ROM tables discussed above has 80 added to its ASCII code to signify that it is the last letter. Assign the 8-byte table to addresses 3F57–3F5E. Construct an associated subroutine entry-address table and assign it to addresses 3F5F–3F62.

Third, write a subroutine to determine the subroutine entry addresses for tokens CE and CF, the tokens for HPOKE and EXG, respectively. The ROM subroutine pointed at by auxiliary RAM control bytes at 012D and 012E serves as a model for writing this subroutine. Assign this subroutine to addresses 3F68–3F76.

"I developed HPOKE primarily for facilitating string packing."

Fourth, specify the auxiliary control bytes at 0134–0138; specify the contents of 0134 to be 02, the number of Basic key words in the new key-word table. The contents of 0135 and 0136 are 3F and 57, respectively, where 3F57 is the first address of the key-word table. The contents of 0137 and 0138 are 3F and 68, where 3F68 is the entry address of the subroutine discussed in Step 3.

Fifth, construct a one-key-word table for EOR in a form similar to the table of the second step. Assign it to bytes at 3F63, 3F64, and 3F65. Form an EOR subroutine entry address and assign it to 3F66 and 3F67.

Sixth, write a subroutine to determine the subroutine entry address for the token FFA2 associated with EOR. Model this subroutine after the one pointed at by auxiliary RAM control bytes at 0132 and 0133. Assign the subroutine to addresses 3F77–3F84.

Seventh, specify the auxiliary RAM control bytes at 0139–013D as 01, 3F, 63, 3F, and 77.

Eighth, write a Basic program using string-packing techniques to store the subroutines written in previous steps. Pack these subroutines into a string. Write another subroutine that transfers these subroutines to protected high RAM. Pack it in the string and enter it via a USR function.

The String Packing Program

The Basic program of the eighth step of the procedure is shown in its initial form in Program Listing 1. Line 40 finds the address of the first byte of the string A$. Lines 50–80 successively read the data of lines 100–270 and POKE them into the 210 available string locations. Data lines 100–270 contain the following information:

- Line 100 contains 10 auxiliary RAM control bytes.
- Line 110 contains a subroutine that transfers those 10 bytes to RAM addresses 0134–013D.
- Line 120 contains a subroutine to transfer the rest of the string information to high RAM.
- Line 130 contains a key-word table and subroutine entry-address table for HPOKE and EXG.
- Line 140 contains a key-word table and subroutine entry-address table for EOR.
- Line 150 contains a subroutine pointed at by bytes assigned to 0137 and 0138.
- Line 160 contains a subroutine pointed at by bytes assigned to 013C and 013D.
- Lines 170–210 contain the subroutine that executes HPOKE.
- Lines 220–250 contain the subroutine that executes EXG.
- Lines 260 and 270 contain the subroutine that executes EOR.

If you own a 32K Color Computer,

```
10 PCLEAR4:PMODE4:COLOR0,5:PCLS:SCREEN1,1
20 X=&H1020
30 HPOKEX,DD,DD,DD,DF:HPOKEX+32,AA,AA,AA,AF:HPOKEX+64,00,00,00,0
7:HPOKEX+96,AA,AA,AA,AF:HPOKEX+128,DD,DD,DD,DF
40 FORI=1TO15:NEXT
50 FORI=X TOX+128STEP32:HPOKEI,FF,FF,FF,FF:NEXT
60 X=X+2
70 IFX=4158THEN20ELSE30
```

Program Listing 2

```
10 PCLEAR4:PMODE4:COLOR0,5:PCLS:SCREEN1,1
20 DIMA(3),C(3)
30 DRAW"BM0,2E2F2E2F2E2F2E2F2E2F2E2F2E2F2G2H2G2H2G2H2G2H2G2H2G2H
2G2H2R28"
40 GET(0,0)-(31,4),A,G
50 GET(0,5)-(31,9),C,G
60 X=0
70 PUT(X,0)-(X+31,4),A,AND
80 FORI=1TO15:NEXT
90 PUT(X,0)-(X+31,4),C,PSET:X=X+16
100 IFX=240THEN60ELSE70
```

Program Listing 3

```
10 CLS:A=&H2AF7:B=&H34C8
20 C=EOR(A,B):PRINTHEX$(C)
30 IFC<>7743THENPRINT"THE EXAMPLE WORKS INCORRECTLY."
```

Program Listing 4

```
10 CLS
20 A$(0)="FIRST":A$(1)="SECOND":A$(2)="THIRD":A$(3)="FOURTH":A$(
4)="FIFTH"
30 FORI=0TO4:PRINTA$(I):NEXT:PRINT
40 FORI=0TO3:EXG(A$(I),A$(I+1)):NEXT
50 FORI=0TO4:PRINTA$(I):NEXT
```

Program Listing 5

change the &H3Fs in lines 100 and 120–160 to &H7Fs.

If you own a 16K Color Disk System you must make the following changes: In line 110 replace the second &H34 with &H3E. Also in line 110, replace the last item &H43 with &H4D. In line 150 replace &HD0 and &HCE with &HE3 and &HE1, respectively. In line 160 replace each &H44 with &H50.

Programs that add key words to the Basic vocabulary have one restriction. You must not use the first three USR functions—USR0, USR1, and USR2—after those programs have been executed. This is no real hardship because the addition of key words reduces the need for USR functions, and you may add to the Basic vocabulary a larger set of USR-like functions of your own. The restriction applies only to Extended Color Basic and *not* to Disk Basic.

Carefully type in Program Listing 1, save it on tape or disk, and run it.

After the program has been executed, there is no need for any of the lines except 30 and 40. Therefore, delete line 20 along with lines 50–270. Then augment the program by adding the following lines:

"If you have a Color Computer, change line 10 to CLEAR200,&H7F57."

```
10 CLEAR200, & H3F57
50 DEFUSR=X+10
60 X=USR(0)
70 NEW
```

If you have a 32K Color Computer, change line 10 to CLEAR200,&H7F57. The augmented program is its final form. Save it on tape or disk before you run it.

Some Examples

These example programs illustrate HPOKE, EXG, and EOR. If they do not work as described, you may not have keyed in Program Listing 1 correctly.

Program Listing 2 is entitled Caterpillar. It moves a caterpillar-like graphics creature across the screen. I developed HPOKE primarily for facilitating string packing, but Program Listing 2 shows HPOKE's utility in the composition of economical graphics programs, using only 238 bytes.

Program Listing 3 is similar to Listing 2. It uses the Color Computer's Draw, Get, and Put statements to move the caterpillar across the screen, but at a cost of 330 bytes.

The HPOKE graphics program requires only 72 percent of the bytes needed using the conventional graphics statements. This example indicates that you may expect potentially high savings using HPOKE for large graphics programs.

Program Listing 4 prints 1E3F on the screen.

Program Listing 5 prints on the first five screen lines the words first, second, third, fourth, and fifth, respectively. The words second, third, fourth, fifth, and first should appear on the seventh to eleventh lines of the screen. ■

H. Allen Curtis is self-employed and has degrees from William and Mary, MIT, and Harvard. He lives at 172 Dennis Drive, Williamsburg, VA 23185.

Games in Color

by Robert Toscani

These games for the Color Computer will test you, amuse you, and frustrate your friends!

Dave McGlumphy's program that imitated the hand-held game, Simon (*80 Micro,* August 1981), is a natural for the Color Computer. I rewrote his program to fit my 16K Extended Color Basic computer, but I have written a 4K version as well.

I often let print statements run together, and to prevent words from being cut in half, I add and remove spaces as needed. So, my lines may have long spaces or no space where one should be, but this keeps the text neat. If necessary, you could compress many of the lines to save memory.

In my version of Simon, lines 45-55 are the color test. Line 60 tells you which letter to push for each color.

Line 75 controls time. The higher the difficulty level, the less time a color is shown, and the less time you have to press buttons. Line 80 determines if the color pattern repeats with each new color or not.

Line 185 is the base line for the 12-piece color sequence. Line 190 randomizes the color selections and line 200 starts the game. The 50-Y*9 at the end is length of time the color appears.

Line 195 compares the answers in the right order, as will be shown later. Lines 210-394 present each color while the IFX= lines determine when to stop.

Lines 400-405 set the screen to black, clear the computer for your answer, and set up the timing loop. Line 410 gives the letter answer a numerical value for comparison in lines 430-540. This is where N comes in, to make sure you're comparing the right color question and letter answer.

If you've made an error, lines 570 and 900-910 tell you, and if you've run out of time before pressing the button, lines 950-980 let you know.

Line 800 plays and shows your answer. If it's correct, lines 810 and 815 determine if you have more buttons to press in that run. If the sequence is up to four colors, you have to press four buttons. Press all four in the correct order and you move up one to five colors. Lines 850 and 860 end the game when the maximum 12-color sequence has run and determine which of the winning answers you get.

These winning answers are in lines 1000-1170 along with some music and the option of playing again or ending the program.

Takeaway

The second game I altered is Richard Ramella's Tanterian Takeaway, from *80 Micro*, December 1981. The program runs on the Color Computer without modification, but I added graphics and wrote another version for two players.

In both versions, you take turns moving from one to ten jewels from one box to another. The winner is the one who moves the 100th jewel, and in the one-person game, the computer always wins. I'll explain why later, but first I'll point out the changes I've made in the one-person program.

Up to line 85, the programs are almost identical. Line 100 starts the graphics. From 100-170, the two boxes are set with the blue one containing 100 jewels. Lines 180-240 remove the jewels from the blue box and place them in the red box. Line 260 determines whose turn it is.

Line 275 starts the game proper and the rest of the program controls it. Most of the listing is self-explanatory, except for the 11-X moves the computer makes. This is the key to the computer's winning. The computer will always move first with one and from then on will take your move and add on enough to move up 11. For example, the computer moves one, you move seven for a total of eight, and the computer moves four for a game total of 12. Then you move five and the computer moves six for a total of 23. But in each case, the computer moved up by 11.

After playing the game a few times, you'll see the computer always ends on the same numbers: 1, 12, 23, 34, 45, 56, 67, 78, 89, and 100. Each number is 11 higher than the last. Realizing this, I knew a two-person game could be written that would allow either person to move first, but would let me win because I knew the key numbers. I could get onto them at some point and from then on move up by 11.

With that in mind, I wrote the program that you can play with your friends and win, as long as they don't know the key numbers. Since the program picks the starter at random, I suggest you keep your beginning moves random and not get onto the key sequence until at least half the jewels are moved. This way, your opponent will be quite mystified by your constant winning.

My program, My Owner, begins by explaining the rules. Line 80 picks a number that decides who starts. Lines 100-240 are almost the same as the other program. They set up the boxes and lights and move them in accordance with the number picked. These lines have some differences from the previ-

The Key Box

Color Computer
16K RAM
Extended Color Basic

QUALITY PERFORMANCE...
...PROFESSIONAL RESULTS

TRS-80 FLOPPY/HARD DISK SOFTWARE

RECORDS MANAGEMENT
File Card

One easy-to-use program replaces your filing cabinet and its contents! No need to keep changing between word processor, database manager, mailing list, file index, or desk calculator. We put it all together in FILE CARD.

Your database entries and related texts appear on screen together with many options for sorts, global searches, reports, and labels. FILE CARD frees the user to dream creatively and work efficiently. Typical applications are:

- Personnel/case records
- Patient/client histories
- Library indexes and abstracts
- Art collection commentaries
- Customer/sales activities
- Product quality profiles
- Vendor information

Model II/16 64K, 2 Disk $224.50

DATA MANAGEMENT
Stat-to-Plotter

Use your multipen intelligent plotter from Radio Shack or Houston Instruments to produce publication quality graphics. Numerical entries from database, datafile, or keyboard are passed through Radio Shack's Statistical Analysis Programs. The results are automatically plotted on paper or projection transparencies. The graphs and legends are accurate enough for scientists and engineering presentations.

Model II/16 64K, 2 Disk $99.50

PROJECT MANAGEMENT
Gantt-Pack V 3.0

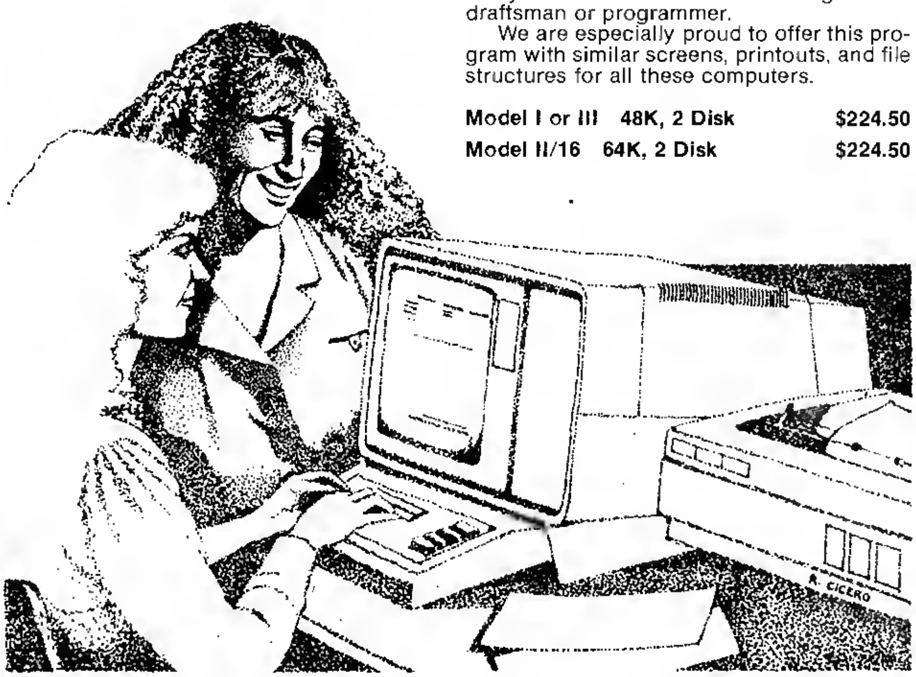
This *FAST*, flexible, powerful program can manage all your scheduling needs. Use the GANTT-PACK charts to visualize:

- Hourly client/patient visits
- Daily appointments and meetings
- Weekly work crew assignments
- Monthly short course offerings
- Yearly research projects

The GANTT-PACK charts and tables help manage time and organize activities. We automated and enhanced the information available on the charts, and made them very easy to update or change. Engineers have used this technique for over fifty years. Now everyone can benefit without having to be a draftsman or programmer.

We are especially proud to offer this program with similar screens, printouts, and file structures for all these computers.

Model I or III 48K, 2 Disk $224.50
Model II/16 64K, 2 Disk $224.50



SPECIALIZED SOFTWARE

HI-RES PRINTING WITH
EP-Graphics

Add our graphics commands to BASIC. By creating a magic slate in memory, you can now print HI-RES charts and graphs. The printouts are full-size printer/plotter emulations. Requires user to write short BASIC programs and have EPSON MX-80 or 100 with Graftrax.

Model I or III 48K, 1 Disk $79.50

CREATE A DISK LIBRARY
Catalog

"Where did I put that program?" With CATALOG, a few keystrokes will find any program/file in your collection. The screens, sorts, printouts, and ability to handle "funny" disks make this the professional's choice.

Model I 48K, 1 Disk $69.50

EDUCATION ASSISTANCE
Tumble

Explore the realism of rotating images with smooth, real-time graphics. Adjust speeds and tilts from either keyboard or joystick with this simple, fun, educational package.

Model I or III 48K, 1 Disk $39.50

SERVICE, QUALITY AND RELIABILITY FOR:

Small businesses, professionals, government contractors, major corporations, schools and colleges, and people from all walks of life.

LIBRA LABORATORIES, INC.
495 MAIN STREET, METUCHEN, NEW JERSEY 08840

201-494-2224

Dealer Inquiries Invited:

Terms:
Orders shipped UPS free for M/C, VISA, check, or money order.
Orders shipped UPS C.O.D. for purchase order number.
International orders add 10%.
Please specify computer model.

Radio Shack Trademark of Tandy Corporation.
TRS-80 Trademark of Tandy Corporation.
Epson Trademark of Shinshu Seiki Co., Ltd.
HI-PLOT Trademark of Houston Instruments. ✓131

ous program, so it's best to type them in anew.

The same applies to lines 250–500. The value of Q determines whose turn it is to move and sends the program to the proper line.

Lines 500 and up are the ending remarks. On the off chance you lose, I've enclosed the appropriate comments. You can also play again without letting the program end.

So, you now have two programs with which to astonish friends and make enemies. The first program has one bug. When the computer makes the first move of one, a dot appears in the red box, but one dot does not wink out in the blue. But, the first dot disappears from the blue box, along with the others, as soon as the next move is made.

Math Test

The third program is a math test. Lines 20 and 40 let you pick the numbers that provide the computer with the numbers for the question pool.

Line 50 lets you pick addition, subtraction, multiplication, or division as the math to be tested on. In subtraction, if the second number is larger than the first, the answer will be negative. To prevent that, answer N to line 55.

Line 60 sets the time limit per problem in seconds, while line 70 lets you pick the total number of problems to be asked and even cut the test short if you answer a given number of questions correctly. The time limit should be no more than 90 to prevent problems with the display.

Line 85 keeps a count of the total questions asked, and line 120 starts the timer. Line 110 asks the question.

Line 121 prints a red bar along the bottom as a visual indication of the time. It decreases by one for each second that ticks off, which is line 132. Also, when half the time has passed, lines 126 and 142 come into play, flashing the message, "Hurry," across the screen.

Lines 122, 125, and 128 set the computer to receive your answer. Line 130 prints it, along with the time, and line 145 completes the loop, enabling the computer to count down. Depending on the moment you press the key, the number may not appear for another second. The computer has to loop to the print command again. Don't hold the key down, just press it in a normal manner. A related problem is that you cannot correct a mistake.

After putting in the complete answer, press enter to break the loop.

Line 147 produces the correct answer

Program Listing 1

```
1 REM CC SIMON BY ROBERT TOSCANI, 4744 WHITAKER AVE. PHILA. PA.
19120
10 CLS
20 PRINT@128,STRING$(32,"*"):PRINT@288,STRING$(32,"*"):PRINT@204
,"CC SIMON"
22 PLAY "T3O3L4CFFGAFAGCFFGAL2FL4E"
25 CLS
30 PRINT"THIS GAME IS BASED ON SIMON. THE SCREEN WILL SHOW DIFFE
RENT      COLORS. YOU HAVE TO PRESS THE    RIGHT LETTER IN THE RIG
HT        SEQUENCE."
35 PRINT"WE'LL SHOW THE COLORS AND THE    MATCHING LETTERS NOW."
40 FORZZ=1TO3000:NEXTZZ
45 FORAA=1TO63:FORBB=1TO31
50 CC=INT(AA/8+1)
55 SET(AA,BB,CC):NEXTBB,AA
60 PRINT" G   Y   B   R   W   C   M   O"
65 PRINT"PRESS ANY KEY TO CONTINUE"
70 C$=INKEY$:IF C$="" THEN70
75 INPUT"PICK LEVEL 1(EASY)-5(HARD)";Y
80 INPUT"DO YOU WANT THE COLOR SEQUENCE  TO CHANGE WITH EACH NEW
 COLOR,   Y(HARDER) OR N(EASIER)";B$
85 CLS
90 PRINT"OK, THE GAME WILL START WITH ONECOLOR AND INCREASE BY O
NE WITH  EACH CORRECT ANSWER. WHEN THE   SCREEN GOES BLACK, THAT
'S YOUR  SIGNAL TO BEGIN PRESSING BUTTONS"
95 PRINT"AND REPEAT THE SEQUENCE.WHEN    YOU'RE READY, PRESS ANY
 KEY TO  START."
100 C$=INKEY$:IFC$="" THEN100
185 X=1
190 A=RND(8):D=RND(8):E=RND(8):F=RND(8):H=RND(8):I=RND(8):J=RND(
8):K=RND(8):L=RND(8):P=RND(8):Q=RND(8):S=RND(8)
195 N=1
200 CLS(0):FORZZ=1TO190:NEXTZZ:CLS(A):SOUND40+20*A,50-Y*9
210 IF X=1 THEN 400
230 CLS(D):SOUND40+20*D,50-Y*9
240 IF X=2 THEN 400
250 CLS(E):SOUND40+20*E,50-Y*9
260 IF X=3 THEN 400
270 CLS(F):SOUND40+20*F,50-Y*9
280 IF X=4 THEN 400
290 CLS(H):SOUND40+20*H,50-Y*9
300 IF X=5 THEN 400
310 CLS(I):SOUND40+20*I,50-Y*9
320 IF X=6 THEN 400
330 CLS(J):SOUND40+20*J,50-Y*9
340 IF X=7 THEN 400
350 CLS(K):SOUND40+20*K,50-Y*9
360 IF X=8 THEN 400
370 CLS(L):SOUND40+20*L,50-Y*9
375 IF X=9 THEN 400
380 CLS(P):SOUND40+20*P,50-Y*9
385 IF X=10 THEN 400
388 CLS(Q):SOUND40+20*N,50-Y*9
392 IF X=11 THEN 400
394 CLS(S):SOUND40+20*P,50-Y*9
400 CLS(0):T=0
403 A$=INKEY$:T=T+1:IF T=500-(Y*75) THEN 950
405 IF A$="" THEN 403
410 IF A$="G" THEN Z=1 ELSE IF A$="Y" THEN Z=2 ELSE IF A$="B" TH
EN Z=3 ELSE IF A$="R" THEN Z=4 ELSE IF A$="W" THEN Z=5 ELSE IF A
$="C" THEN Z=6 ELSE IF A$="M" THEN Z=7 ELSE IF A$="O" THEN Z=8
430 IF N=1 AND Z=A THEN 800
440 IF N=2 AND Z=D THEN 800
450 IF N=3 AND Z=E THEN 800
460 IF N=4 AND Z=F THEN 800
470 IF N=5 AND Z=H THEN 800
480 IF N=6 AND Z=I THEN 800
490 IF N=7 AND Z=J THEN 800
500 IF N=8 AND Z=K THEN 800
510 IF N=9 AND Z=L THEN 800
520 IF N=10 AND Z=P THEN 800
530 IF N=11 AND Z=Q THEN 800
540 IF N=12 AND Z=S THEN 800
570 GOTO 900
800 CLS(Z):SOUND40+20*Z,5
810 IF N=X AND B$="N" THEN 850
815 IF N=X AND B$="Y" THEN 860
820 N=N+1:GOTO400
850 X=X+1:IF X=13 THEN 1000 ELSE GOTO 195
860 X=X+1:IF X=13 THEN 1100 ELSE GOTO 190
900 PRINT "WRONG ANSWER, SORRY.";
905 PLAY"T3O3L2CL3CL8CL2CE-L8DL3DL8CL3CO2L8BO3L2C"
```

Listing 1 continues

See List of Advertisers on Page 563

Listing 1 continued

```
910 INPUT"CARE TO TRY AGAIN Y/N";C$:IF C$="Y" THEN 185 ELSE END
950 PRINT"TIME'S UP. YOU LOSE."
960 PLAY"O3L8AAAAL4AGL1GP4L8GGGGL4GFL1F"
980 INPUT"WANT TO TRY AGAIN Y/N";C$:IF C$="Y" THEN 185 ELSE END
1000 PRINT"YOU WIN!!!"
1005 PLAY"O2L8A#O3L4CL8E-L3GL16G-L8GB-O2B-O3L4CL8E-L1G"
1010 PRINT"WHY NOT TRY THE NEXT LEVEL?"
1020 INPUT"PRESS 1 TO MOVE UP, 2 TO STAY AT THE SAME LEVEL AND 3
 TO END";C$
1030 IF C$="1" THEN 100 ELSE IF C$="2" THEN 185 ELSE IF C$="3" T
HEN END
1100 PRINT "TERRIFFIC. YOU'VE WON.";
1110 PLAY"T3O3L8.ODDL2GO4DP8L8.CO3BAO4L2GL4DP8L8.CO3BAO4L2GL4DL8
.CO3BO4CL2O3A"
1140 PRINT"WANT TO TRY AGAIN SINCE YOU'RE"
1150 PRINT"DOING SO WELL? PRESS 1 TO GO "
1160 INPUT"AGAIN, 2 TO CHANGE SETTING AND 3 TO END";C$
1170 IF C$="1" THEN 185 ELSE IF C$="2" THEN 100 ELSE IF C$="3" T
HEN END
```

Program Listing 2

```
1 REM CC SIMON BY ROBERT TOSCANI, 4744 WHITAKER AVE. PHILA. PA.
19120
10 CLS
45 FORAA=1TO63:FORBB=1TO31
50 CC=INT(AA/8+1)
55 SET(AA,BB,CC):NEXTBB,AA
60 PRINT" G    Y    B    R    W    C    M    O"
65 PRINT"PRESS ANY KEY TO CONTINUE"
70 C$=INKEY$:IF C$="" THEN70
75 INPUT"PICK LEVEL 1(EASY)-5(HARD)";Y
80 INPUT"DO YOU WANT THE COLOR SEQUENCE  TO CHANGE WITH EACH NEW
 COLOR,  Y(HARDER) OR N(EASIER)";B$
95 PRINT"PRESS ANY KEY TO START"
100 C$=INKEY$:IFC$="" THEN100
185 X=1
190 A=RND(8):D=RND(8):E=RND(8):F=RND(8):H=RND(8):I=RND(8):J=RND(
8):K=RND(8):L=RND(8):P=RND(8)
195 N=1
200 CLS(0):FORZZ=1TO190:NEXTZZ:CLS(A):SOUND40+20*A,50-Y*9
210 IF X=1 THEN 400
230 CLS(D):SOUND40+20*D,50-Y*9
240 IF X=2 THEN 400
250 CLS(E):SOUND40+20*E,50-Y*9
260 IF X=3 THEN 400
270 CLS(F):SOUND40+20*F,50-Y*9
280 IF X=4 THEN 400
290 CLS(H):SOUND40+20*H,50-Y*9
300 IF X=5 THEN 400
310 CLS(I):SOUND40+20*I,50-Y*9
320 IF X=6 THEN 400
330 CLS(J):SOUND40+20*J,50-Y*9
340 IF X=7 THEN 400
350 CLS(K):SOUND40+20*K,50-Y*9
360 IF X=8 THEN 400
370 CLS(L):SOUND40+20*L,50-Y*9
375 IF X=9 THEN 400
380 CLS(P):SOUND40+20*P,50-Y*9
400 CLS(0):T=0
403 A$=INKEY$:T=T+1:IF T=500-(Y*75) THEN 950
405 IF A$="" THEN 403
410 IF A$="G" THEN Z=1 ELSE IF A$="Y" THEN Z=2 ELSE IF A$="B" TH
EN Z=3 ELSE IF A$="R" THEN Z=4 ELSE IF A$="W" THEN Z=5 ELSE IF A
$="C" THEN Z=6 ELSE IF A$="M" THEN Z=7 ELSE IF A$="O" THEN Z=8
430 IF N=1 AND Z=A THEN 800
440 IF N=2 AND Z=D THEN 800
450 IF N=3 AND Z=E THEN 800
460 IF N=4 AND Z=F THEN 800
470 IF N=5 AND Z=H THEN 800
480 IF N=6 AND Z=I THEN 800
490 IF N=7 AND Z=J THEN 800
500 IF N=8 AND Z=K THEN 800
510 IF N=9 AND Z=L THEN 800
520 IF N=10 AND Z=P THEN 800
570 GOTO 900
800 CLS(Z):SOUND40+20*Z,5
810 IF N=X AND B$="N" THEN 850
815 IF N=X AND B$="Y" THEN 860
820 N=N+1:GOTO400
850 X=X+1:IF X=11 THEN 1000 ELSE GOTO 195
```

Listing 2 continues

to the question. In division questions, the program rounds off the answer to the first decimal point. If the second decimal is 5 or higher, increase the first decimal by one to get the correct answer.

Line 150 compares your answer with the correct one, and lines 160–300 print the appropriate response. Line 302 keeps track of the number of correct answers. Line 306 ends the program if either it or the total number of questions is reached. If they are, lines 400–417 provide a brief sound and light show, and then lines 430–500 tell you how many questions you got right and the percentage of correct answers. Line 440 rounds off the answer to two decimal places and then converts that to a whole number to provide you with the percentage.

Line 505, based on the percentage, chooses which phrase to print from the next nine lines, and then lines 700–729 let you begin again or end the program. Lines 1000 and up are the introduction, explaining the rules of the test. ■

Robert Toscani can be reached at 4744 Whitaker Ave., Philadelphia, PA 19120.

See List of Advertisers on Page 563

Listing 2 continued

```
860 X=X+1:IF X=11 THEN 1100 ELSE GOTO 190
900 PRINT "WRONG ANSWER, SORRY.";
905 SOUND 20,6
910 INPUT"CARE TO TRY AGAIN Y/N";C$:IF C$="Y" THEN 185 ELSE END
950 PRINT"TIME'S UP. YOU LOSE."
960 SOUND80,6
980 INPUT"WANT TO TRY AGAIN Y/N";C$:IF C$="Y" THEN 185 ELSE END
1000 PRINT"YOU WIN!!!"
1005 SOUND 150,10
1010 PRINT"WHY NOT TRY THE NEXT LEVEL?"
1020 INPUT"PRESS 1 TO MOVE UP, 2 TO STAY AT THE SAME LEVEL AND 3
 TO END";C$
1030 IF C$="1" THEN 100 ELSE IF C$="2" THEN 185 ELSE IF C$="3" T
HEN END
1100 PRINT "TERRIFFIC. YOU'VE WON.";
1110 SOUND200,12
1140 PRINT"WANT TO TRY AGAIN SINCE YOU'RE"
1150 PRINT"DOING SO WELL? PRESS 1 TO GO "
1160 INPUT"AGAIN, 2 TO CHANGE SETTING AND 3 TO END";C$
1170 IF C$="1" THEN 185 ELSE IF C$="2" THEN 100 ELSE IF C$="3" T
HEN END
```

Program Listing 3

```
10 CLS
20 PRINT"HUMAN RAPSCALLION!"
30 PRINT@64,"I'VE CAUGHT YOU STEALING THE 100JEWELS OF PLANET TA
NTER."
40 PRINT@160,"ORDINARILY, THE PENALTY IS      SEVERE AND TERMINA
L..."
50 PRINT@256,"BUT I'M A GAMESMAN. FOR YOUR    LIFE AND THE JEWEL
S YOU MUST    MATCH WITH ME AT TANTERIAN      TAKEAWAY. HIT ENTE
R FOR RULES."
60 INPUT A$:CLS
70 PRINT"WE TAKE TURNS MOVING 1 TO 10    JEWELS FROM THE BLUE TO
 THE RED BOX. THE ONE WHO MOVES THE 100THJEWEL IS THE WINNER OF
LIFE AND TREASURE. HIT ENTER TO START."
80 INPUT A$
85 GOTO 275
100 PMODE 3,1:SCREEN 1,0
110 PCLS1
115 COLOR 3,2
120 LINE(17,50)-(17,150),PSET:LINE(17,150)-(123,150),PSET:LINE(1
23,150)-(123,50),PSET
125 COLOR 4,2
130 LINE(140,50)-(140,150),PSET:LINE(140,150)-(250,150),PSET:LIN
E(250,150)-(250,50),PSET
140 FOR A=54 TO 146 STEP 10
150 FOR B=24 TO 118 STEP 10
160 PSET(B,A,2)
170 NEXT B,A
180 C=24:D=54
185 FOR Y=1 TO X
192 SOUND C,1
195 PRESET(24,54)
200 PRESET(C,D)
210 PSET((267-C),(198-D),2)
220 C=C+10:IF C=124 THEN D=D+10:IF C=124 THEN C=24
240 NEXTY
250 FOR Z=1 TO 1100:NEXTZ
260 IF Q=0 THEN 300 ELSE IF Q=1 THEN 400 ELSE IF Q=3 THEN 500
275 CLS
280 PRINT"I GO FIRST, MOVING 1 FOR A TOTALOF 1."
285 FOR Z=1 TO 1000:NEXTZ
287 X=1:Q=0
288 N=1
290 GOTO 100
300 CLS
310 INPUT"HOW MANY ARE YOU MOVING";X
320 IF X<1 THEN 350 ELSE IF X=<10 THEN 370
330 PRINT"THE HUMAN CHEATS! TAKE 10 OR    FEWER!"
340 GOTO 310
350 PRINT"YOU MUST MOVE AT LEAST ONE."
360 GOTO 310
370 PRINT"YOUR TOTAL IS AT "N+X
380 PRINT"BY TAKING "X
385 FOR Z=1 TO 1100:NEXTZ
390 PMODE3,1:SCREEN1,0:Q=1:GOTO 185
400 PRINT"I MOVE "11-X
410 PRINT"FOR A TOTAL OF "N+X+(11-X)
415 FOR Z=1 TO 1100:NEXTZ
418 N=N+X+(11-X)
420 IF N=100 THEN Q=3 ELSE Q=0
425 X=11-X
430 PMODE3,1:SCREEN 1,0:GOTO 185
500 CLS
510 PRINT"I WIN, MY DOOMED FRIEND."
520 PLAY"V25T3O3L2CL3CL8CL2CE-L8DL3DL8CL3CO2L8BO3L2C"
530 PRINT@64,"AND NOW FOR A CONFESSION:"
540 PRINT@128,"THE GAME IS RIGGED SO I ALWAYS  WIN. THIS SHOULD
TEACH YOU TO   NEVER TRUST A TANTERIAN..."
550 FOR Z=1 TO 1500:NEXTZ
560 CLS
570 FOR E=1 TO 320
580 PRINT "HA";
590 NEXT E
600 CLS
610 PRINT"LET'S PLAY AGAIN AND PROLONG    YOUR MISERABLE LIFE. Y
ES OR NO?"
620 INPUT W$
630 IF W$="YES" THEN 275 ELSE CLS
640 PRINT"THAT'S ALL FOLKS"
650 END
```

Program Listing 4

```
1 REM  MY OWNER BY ROBERT TOSCANI, 4744 WHITAKER AVE. PHILA. PA.
 19120
10 CLS
20 PRINT"YOU ARE ABOUT TO MATCH WITS WITHMY OWNER. I WARN YOU, M
```

Listing 4 continues

Listing 4 continued

```
Y OWNER  IS VERY SMART AND WILL PROBABLY BEAT YOU."
30 PRINT"THE RULES OF THE CONTEST ARE    VERY SIMPLE. WE HAVE A
BLUE BIN FILLED WITH 100 LIGHTS. TAKING  TURNS EACH OF YOU WILL
MOVE FROM1 TO 10 LIGHTS OVER TO THE RED  BOX."
40 PRINT"WHOEVER MOVES THE 100TH LIGHT   WINS!"
50 INPUT"IF YOU'RE READY, PLEASE ENTER   YOUR NAME";B$


55 CLS
60 PRINT"OKAY, I WILL NOW PICK A NUMBER  AT RANDOM. IF IT IS ODD
, "B$
70 PRINT"WILL GO FIRST. EVEN AND MY OWNER STARTS."
75 FOR Z=1 TO 1100:NEXT Z
80 U=RND(10):IF U=1 OR U=3 OR U=5 OR U=7 OR U=9 THEN90
85 PRINT"THE NUMBER IS "U" SO MY OWNER   BEGINS.":FOR Z=1 TO 110
0:NEXTZ:GOTO 100
90 PRINT"THE NUMBER IS "U" SO YOU START.":FOR Z=1 TO 1100:NEXT Z
100 PMODE3,1:SCREEN1,0


110 PCLS1
115 COLOR 3,2
120 LINE(17,50)-(123,150),PSET,BF
125 COLOR 4,2
130 LINE(140,50)-(250,150),PSET,BF
140 FOR A=54 TO 146 STEP 10
150 FOR B=24 TO 118 STEP 10
160 PSET(B,A,2)
170 NEXT B,A
175 FOR Z=1 TO 300:NEXTZ
180 C=24:D=54
181 IF U=1 OR U=3 OR U=5 OR U=7 OR U=9 THEN 300 ELSE GOTO 399
182 IF Q=1 OR Q=3  THEN X=N ELSE X=H
185 FOR Y=1 TO X
192 SOUND C,1
195 PSET(C,D,3)


210 PSET((267-C),(198-D),2)
220 C=C+10:IF C=124 THEN D=D+10:IF C=124 THEN C=24
240 NEXTY
250 FOR Z=1 TO 1100:NEXTZ
260 IF Q=0 THEN 300 ELSE IF Q=1 THEN 399 ELSE IF Q=3 THEN 500 EL
SE IF Q=4 THEN 530
300 CLS


310 INPUT"HOW MANY ARE YOU MOVING";N
320 IF N<1 THEN 350 ELSE IF N=<10 THEN 370
330 PRINT"TAKE 10 OR LESS."
340 GOTO 310
350 PRINT"YOU MUST MOVE AT LEAST ONE."
360 GOTO 310
370 PRINT"YOU MOVE "N
375 R=N+R
380 PRINT"THAT MAKES THE TOTAL MOVED "R
385 FOR Z=1 TO 1100:NEXTZ
387 IF R=100 THEN Q=3 ELSE Q=1
390 PMODE3,1:SCREEN1,0
395 GOTO 182
399 CLS
400 INPUT"HOW MANY DOES MY OWNER WANT TO  MOVE";H

401 IF H>0 AND  H<=10 THEN 404
402 PRINT"MY OWNER JOKES WITH YOU THAT IS AN ILLEGAL MOVE AS MY
OWNER     KNOWS."
403 GOTO 400
404 R=H+R
410 PRINT"MY OWNER MOVES "H
415 PRINT"FOR A TOTAL OF "R
418 FOR Z=1 TO 1100:NEXT Z
420 IF R=100 THEN Q=4 ELSE Q=0
430 PMODE3,1:SCREEN 1,0
440 GOTO 182
500 CLS
510 PRINT B$" WINS?!?!?!"
520 PRINT" AW, YOU JUST GOT LUCKY."
530 P$="T4O4L4CO3L4.BL8O4DL4O3B-L2B-L4B-L4.AL8FL4AL1BP4"
532 S$="O4L4CL4.O3GL8EL4DL2DL4DL4.CL8DL4EL1FP4"
534 T$="L4O3EL4.EL8EL4DDDDL1C"
540 PLAY P$:PLAY P$
542 IF Q=4 THEN 550
545 GOTO600
550 CLS
560 PRINT"MY OWNER WINS!!! HA HA. I TOLD  YOU  MY OWNER WAS TOO
SMART FOR YOU."
600 PLAY S$:PLAY T$
605 CLS
610 PRINT"CARE TO, UH, TRY YOUR LUCK AGAIN "B$"?"
615 PLAY S$
620 PRINT"MAYBE THE RESULTS WILL BE A     LITTLE DIFFERENT NEXT
TIME. YES OR NO?"
625 PLAY T$
630 INPUT W$: IF W$="NO" THEN 640
635 R=0:GOTO 60
640 PRINT"THAT'S ALL FOLKS"
650 END
```

Program Listing 5

```
1 REM MATH QUIZ BY ROBERT TOSCANI, 4744 WHITAKER AVE. PHILA. PA.
 19120
9 GOSUB 1000
10 CLS
12 INPUT "WHAT'S YOUR NAME";N$
20 INPUT"PICK A NUMBER";A
40 INPUT"PICK ANOTHER";B
50 INPUT"DO YOU WANT TO ADD(+), SUBTRACT (-), MULTIPLY(*), OR DI
VIDE(/)";Z$
52 IF Z$="+" OR Z$="-" OR Z$="*" OR Z$="/" THEN 55 ELSE PRINT"MU
ST BE ONE OF THOSE SIGNS"
53 GOTO50
55 INPUT"ARE NEGATIVE NUMBERS OK Y/N";Y$
```

Listing 5 continues

Listing 5 continued

```
60 INPUT"WHAT TIME LIMIT PER PROBLEM";Z
65 U=0:I=0
70 INPUT"TOTAL NUMBER OF PROBLEMS";P:INPUT"AND CORRECT ANSWERS B
EFORE IT   ENDS";O
75 PRINT"PRESS ENTER TO BEGIN. GOOD LUCK":INPUT A$
80 CLS
85 I=I+1
90 X=RND(A):Y=RND(B)
100 IF Y$="N" AND X<Y THEN 90
110 PRINT@8,"WHAT IS "
111 X$=STR$(X)
112 PRINT@17,X:PRINT@18+LEN(X$),Z$:PRINT@19+LEN(X$),Y
120 T=Z
121 FOR L=352 TO 351+Z:PRINT@L,CHR$(143+48);:NEXTL
122 R$=""
125 W$=INKEY$
126 IF T=<(Z+1)/2 THEN PRINT@448,"       HURRY!!!   ";
128 R$=R$+W$
130 PRINT@40,R$:PRINT@470,"TIME "T
132 PRINT@L,CHR$(143);:L=L-1
135 T=T-1:SOUND128,1
138 FOR Q=1 TO 275: NEXT Q
140 IF T=<-1 THEN 147
142 IF T=<(Z+1)/2 THEN PRINT@448,"                    ";
144 IF W$=CHR$(13) THEN 147
145 IF T>-1 THEN 125
147 IF Z$="+" THEN D=X+Y ELSE IF Z$="-" THEN D=X-Y ELSE IF Z$="*
" THEN D=X*Y ELSE IF Z$="/" THEN D=INT(((X/Y)+.05)*10)/10
149 IF VAL(R$)=D AND T=<-1 THEN 205 ELSE IF VAL(R$)=D THEN 300
160 PRINT@73,"WRONG, "N$
165 GOTO 210
205 PRINT@73, "TIME'S UP "N$
207 IF VAL(R$)=D THEN 300
210 PRINT@102,"THE ANSWER IS "D
220 GOTO 305
300 PRINT@137,"CORRECT, "N$
302 U=U+1
305 PRINT@161,"THAT'S "U" OUT OF "I" CORRECT"
306 IF U=O OR  I=P THEN 400
307 PRINT@226,"TO TRY AGAIN, PRESS (ENTER)."
320 INPUT A$:GOTO 80
400 FOR W=1 TO 460:NEXT W
401 G=0
402 Q=RND(8):S=RND(255)
410 CLS(Q):SOUND S,1
416 G=G+1
417 IF G<12 THEN 402
430 PRINT @ 3, "THAT'S "U" RIGHT OUT OF"
432 PRINT @ 35, I"QUESTIONS."
440 V=(INT(((U/I)+.005)*100)/100)*100
500 PRINT@67,"THAT'S "V"% CORRECT."
505 IF V=100  THEN 510 ELSE IF V=>80 THEN 560 ELSE IF V=>50 THEN
 605 ELSE IF V<50 THEN 660
510 PRINT@99,"EXCELLENT, "N$
520 GOTO 700
560 PRINT@99, "VERY GOOD, "N$"."
570 GOTO 700
605 PRINT@99,"COULD BE BETTER, "N$"."
620 GOTO 700
660 PRINT@99,"THAT'S TERRIBLE, "N$"!"
665 SOUND 45, 15
670 PRINT@162,"TRY AGAIN AND SEE IF YOU CAN    DO BETTER!";
700 PRINT@226,"TO TRY AGAIN PRESS 1, TO LET    SOMEONE ELSE TRY,
 PRESS 2 AND   TO END, PRESS 3"
710 INPUT Q
720 IF Q=1 THEN 20 ELSE IF Q=2 THEN 10 ELSE IF Q=3 THEN END
1000 CLS
1010 PRINT"THIS IS A MATH QUIZ. (BOO,HISSS) HOWEVER, YOU HAVE QU
ITE A FEW  CHOICES. YOU WILL PICK THE      LARGEST NUMBERS FROM
WHICH THE  QUESTIONS WILL BE DRAWN. YOU    CHOOSE  WHAT TYPE OF
MATH TO BE DRILLED ON. YOU SET THE TIME    LIMIT FOR
1020 PRINT"EACH PROBLEM, HOW MANY TOTAL    QUESTIONS THERE WILL
BE AND HOW MANY CORRECT ANSWERS YOU MUST   HAVE TO END. YOU CAN
ALSO DECIDE IF NEGATIVE NUMBERS WILL BE    USED IN SUBTRACTION."
1030 INPUT"PRESS ENTER TO CONTINUE";A$
1040 CLS
1050 PRINT"ALL DECIMALS MUST BE ROUNDED OFF TO THE FIRST PLACE.
EX. 2.34   WOULD BE 2.3 BUT  2.38 WOULD BE 2.4 IF THE SECOND DEC
IMAL IS 5  OR OVER THEN INCREASE THE FIRST  DECIMAL BY ONE."
1060 PRINT"ONCE YOU HAVE PUT THE ANSWER IN, PRESS ENTER."
1065 INPUT"PRESS ENTER TO CONTINUE";A$
1067 CLS
1070 PRINT"ALSO KNOW THAT ONCE ANYTHING    APPEARS, IT CAN'T BE
CHANGED. IF YOU PRESS THE WRONG KEY, YOU  CANNOT CHANGE THE ANSW
ER."
1075 FOR Q=1TO1000:NEXTQ:CLS
1080 PRINT"ONE FINAL POINT. IT MAY TAKE A  SECOND OR TWO FOR THE
 ANSWER TO APPPEAR. DON'T WORRY, JUST BE   CAREFUL AND WAIT."
1090 INPUT"PRESS ENTER TO CONTINUE";A$
1100 RETURN
```

Program Listing 6

```
1 REM MATH QUIZ BY ROBERT TOSCANI, 4744 WHITAKER AVE. PHILA. PA.
 19120
10 CLS
12 INPUT "WHAT'S YOUR NAME";N$
20 INPUT"PICK A NUMBER";A
40 INPUT"PICK ANOTHER";B
50 INPUT"DO YOU WANT TO ADD(+), SUBTRACT (-), MULTIPLY(*), OR DI
VIDE(/)";Z$
52 IF Z$="+" OR Z$="-" OR Z$="*" OR Z$="/" THEN 55 ELSE PRINT"MU
ST BE ONE OF THOSE SIGNS"
53 GOTO50
55 INPUT"ARE NEGATIVE NUMBERS OK Y/N";Y$
60 INPUT"WHAT TIME LIMIT PER PROBLEM";Z
65 U=0:I=0
70 INPUT"TOTAL NUMBER OF PROBLEMS";P:INPUT"AND CORRECT ANSWERS B
EFORE IT   ENDS";O
75 PRINT"PRESS ENTER TO BEGIN. GOOD LUCK":INPUT A$
80 CLS
85 I=I+1
90 X=RND(A):Y=RND(B)
100 IF Y$="N" AND X<Y THEN 90
110 PRINT@18,"WHAT IS "
```

Listing 6 continues

Listing 6 continued

```
111 X$=STR$(X)
112 PRINT@17,X:PRINT@18+LEN(X$),Z$:PRINT@19+LEN(X$),Y
120 T=Z
122 R$=""
125 W$=INKEY$
128 R$=R$+W$
130 PRINT@40,R$:PRINT@470,"TIME "T
135 T=T-1:SOUND128,1
138 FOR Q=1 TO 375:NEXT Q
140 IF T=<-1 THEN 147
144 IF W$=CHR$(13) THEN 147
145 IF T>-1 THEN 125
147 IF Z$="+" THEN D=X+Y ELSE IF Z$="-" THEN D=X-Y ELSE IF Z$="*
" THEN D=X*Y ELSE IF Z$="/" THEN D=INT(((X/Y)+.05)*10)/10
149 IF VAL(R$)=D AND T=-1 THEN 205 ELSE IF VAL(R$)=D THEN 300
160 PRINT@73,"WRONG, "N$
165 GOTO 210
205 PRINT@73, "TIME'S UP "N$
207 IF VAL(R$)=D THEN 300
210 PRINT@102,"THE ANSWER IS "D
220 GOTO 305
300 PRINT@137,"CORRECT, "N$
302 U=U+1
305 PRINT@161,"THAT'S "U" OUT OF "I" CORRECT"
306 IF U=O OR  I=P THEN 400
307 PRINT@226,"TO TRY AGAIN, PRESS (ENTER)."
320 INPUT A$:GOTO 80
400 FOR W=1 TO 460:NEXT W
401 G=0
402 Q=RND(8):S=RND(255)
410 CLS(Q):SOUND S,1
416 G=G+1
417 IF G<12 THEN 402
430 PRINT @ 3, "THAT'S "U" RIGHT OUT OF"
432 PRINT @ 35, I"QUESTIONS."
440 V=(INT(((U/I)+.005)*100)/100)*100
500 PRINT@67,"THAT'S "V"% CORRECT."
505 IF V=100  THEN 510 ELSE IF V=>80 THEN 560 ELSE IF V=>50 THEN
 605 ELSE IF V<50 THEN 660
510 PRINT@99,"EXCELLENT, "N$
520 GOTO 700
560 PRINT@99, "VERY GOOD, "N$"."
570 GOTO 700
605 PRINT@99,"COULD BE BETTER, "N$"."
620 GOTO 700
660 PRINT@99,"THAT'S TERRIBLE, "N$"!"
665 SOUND 45, 15
670 PRINT@162,"TRY AGAIN AND SEE IF YOU CAN     DO BETTER!";
700 PRINT@226,"TO TRY AGAIN PRESS 1, TO LET     SOMEONE ELSE TRY,
 PRESS 2 AND   TO END, PRESS 3"
710 INPUT Q
720 IF Q=1 THEN 20 ELSE IF Q=2 THEN 10 ELSE IF Q=3 THEN END
1060 PRINT"ONCE YOU HAVE PUT THE ANSWERE IN, PRESS ENTER."
1067 CLS
```


✓See List of Advertisers on Page 563

SURPRISE! 3-D

It had to happen—sooner or later someone was going to write a stereoscopic 3-D program for the Color Computer. The following two programs are *80 Micro*'s pioneering efforts in what we hope will become a popular programming technique.

Our efforts are by no means spectacular; our intentions are to present the principles involved in writing 3-D programs with examples.

Dennis Kitsz, who has long been a guru for Color Computer users, gives a good discussion on how the illusion of 3-D works. His program draws a stack of blocks on the screen—actually two angles of the same stack superimposed.

Our resident CC wiz, Jake Commander, used the same basic concepts, but added motion. His program displays a rotating cube. We're sure you'll find his techniques fascinating.

As the authors point out, you must be able to get a good picture on the monitor, and the color and contrast adjustments must be precise. And, of course, you must have a color monitor. If you have trouble getting the 3-D effect on your monitor, try a friend's.

Remember, though, that these programs give only the illusion of 3-D, just as the 3-D movies of the 1950s created only illusions. Some sections of the 3-D figures in the programs might appear to "come out" more than others.

We Are Not Alone

Like any good idea, someone else is bound to think of it, too. Well, that happened with the 3-D Color Computer programs. The November 1982 issue of *The Rainbow*, a magazine for the Color Computer, features a 3-D arcade game called Star Trench Warfare by Fred Scerbo. *The Rainbow*'s publisher, Lonnie Falk, said he isn't going to have the bind-in 3-D glasses, though. The glasses found elsewhere in this issue will work with that program.

We welcome *The Rainbow*'s efforts in 3-D programming, and we suggest you pick up their November issue if 3-D interests you. *The Rainbow*'s address is 5803 Timber Ridge Drive, P.O. Box 209, Prospect, KY 40059, (502) 228-4492.

We hope to generate a whole new genre of Color Computer programs using 3-D, but it's up to you readers to refine this technique. If you come up with any ideas, improvements, or interesting applications for 3-D programs, let us know. ■

3-D for Real

by Dennis Bathory Kitsz

Until now, "3-D" has been a misnomer for CC graphics. Our CC expert has done the groundwork for real 3-D. It'll come out and grab you!

Simulated three-dimensional imagery from two dimensions is fascinating, but I found the prospect of programming it intimidating. I am far from being a mathematician, so high-powered, three-dimensional rotation algorithms are simply outside my scope. Instead, I can handle a camera well enough, and can play around in my darkroom. I decided this could be put to work to learn more about 3-D principles.

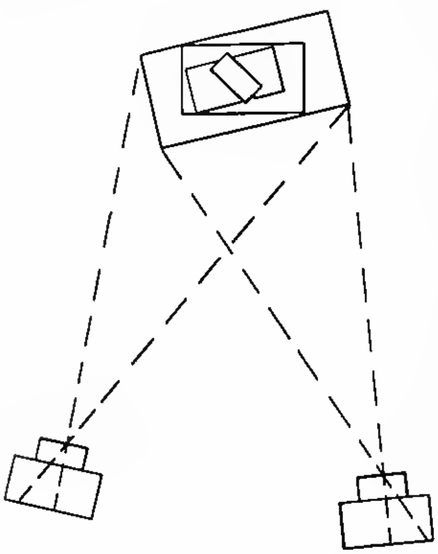


Fig. 1. Camera setup for stereoptical photo. Although any pair of photos taken 3 to 4 inches apart will work in stereo, most impressive results are obtained with longer depth of field, with different objects at several distances from the lens.

The Key Box
Color Computer 16K RAM Extended Color Basic

My idea was to back into 3-D computer graphics by first creating the photographic images—"Vu-Master" style—and rendering these into simple computer versions. The steps I chose were straightforward:

- Select the subject to be examined.
- Photograph it from the desired distance.
- Photograph it again from a short distance to one side.
- Develop and print the separate photos.
- Create and position overlay films.
- Determine X-Y coordinates for each photo.
- Enter the information into the computer.

The original subject matter was a small pile of wooden pieces (Photos 1a and 1b), with pleasantly sharp angles. The principle of three-dimensional photography is well known: each eye views a subject from a slightly different angle, and these images are resolved by the brain into a sense of depth. By providing individual and slightly different flat photographic images to each eye, an illusion of 3-D can be created.

To view Photos 1a and 1b in 3-D, place the magazine comfortably in front of you and in good light, relax your eyes and focus into the distance; move your eyes slowly down to the images in the magazine. It is essential to relax, or else you will see only one of the two images. Once your eyes "float" apart, begin to focus slowly until the images are clear, but still overlap. You can improve this effect by viewing these photos simultaneously through separate magnifying eyepieces, so each eye is forced to see a unique view.

This technique was used for novelty photographs especially at the turn of the century, and is still used in Vu-Master

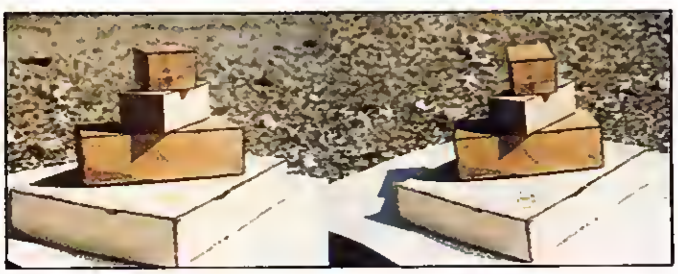


Photo 1. Stereoptical view of a pile of wooden blocks for separated eyes. (a) Left eye. (b) Right eye. Ektachrome 64, natural lighting, f/16, 1/1000 second.

and other stereoptical viewers. It is employed for the photographing of extremely fragile museum specimens. Four pairs of views—top, bottom, and two sides—are usually taken, and the results examined with a simple viewer. This provides a detailed, perspective view so interested visitors don't handle the original specimen.

The trick, however, is to produce a single photograph that appears in three dimensions. Somehow the images must be separated, drawn off the page, and relegated one to the left eye and the other to the right. A composite image is therefore created that can be printed in two distinct (actually complementary) colors, and can be viewed through special lenses.

The colors are usually green and red. When the red printing is viewed through the red lens, it effectively disappears; the green printing looks black through the red lens. Likewise, when green printing is viewed through the green lens, it disappears; the red printing looks black through that lens.

"The trick. . . is to produce a single photograph that appears in three dimensions."

Again, the two-color process has been used for novelty photography in books and magazines, as well as for special effects in films. For readers old enough to remember, it enjoyed brief popularity in 1950s horror films, and is undergoing a small renaissance today.

A three-dimensional computer image would use the same process by drawing on a single screen two slightly different pictures in complementary colors. You can use three-dimensional algorithms to draw directly from design information, a tablet or light pen for input, or you can program the coordinates of an ex-

Photo 2. Stereoptical view of David Gunn, for crossed eyes. (a) Right eye. (b) Left eye. David had to hold the pose carefully since only one camera was used and had to be moved between shots; nevertheless, he did raise one eyebrow. Ektachrome 64, natural lighting, f/22, 1/1000 second.

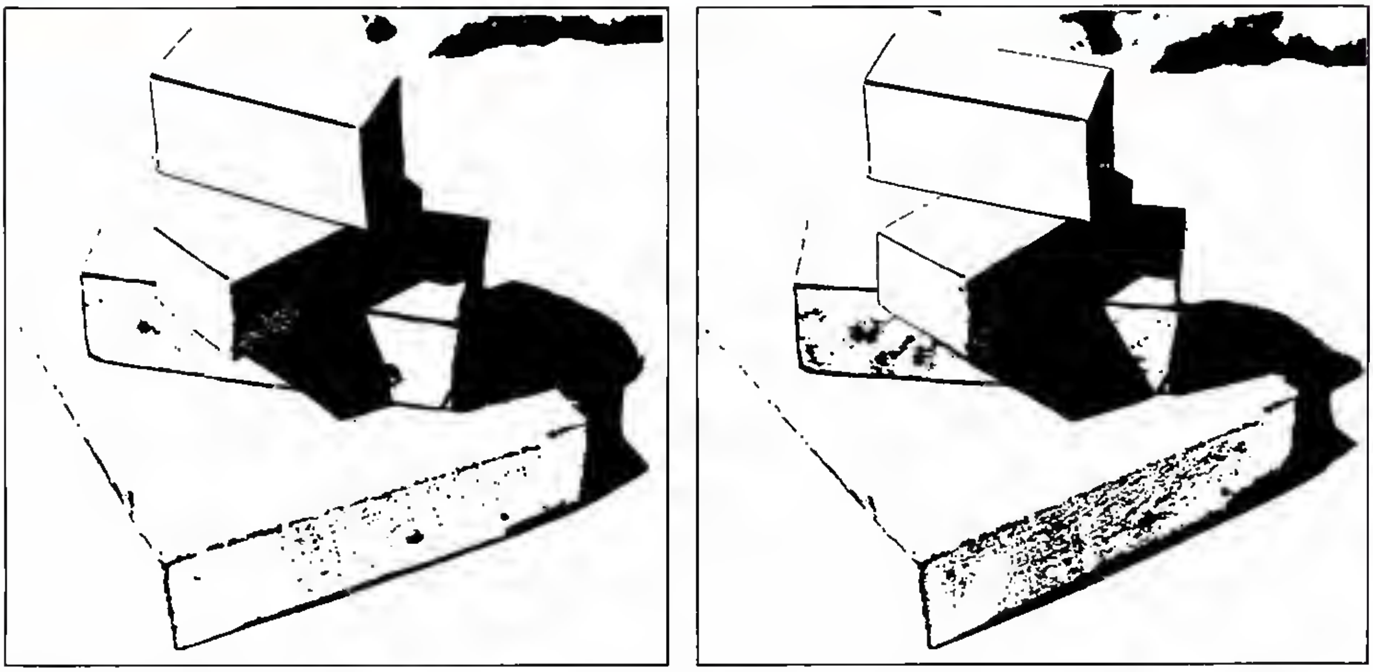


Photo 3. High-contrast film stereo view of the blocks in Photo 1. (a)Left eye. (b)Right eye. Kodalith, natural lighting, ⊥/2.8, 1/15 second, printed on Kodabrome II ultra-hard paper developed in graphic arts developer.

M-ZAL™

Release Two Is Here!

Our Users asked for it, and we made the best TRS-80 disk assembler even better!
Now M-ZAL has:

- Recursive MACROS with full symbolic parameter substitution
- Conditional assembly with unlimited nesting
- Text Editor warm-start/recovery option
- Support for all popular source file formats
- Load Module inspect/superzap using LINKER

and the features that made it famous:

- True multi-pass assembly
- Full Screen Option Menus
- ENTRY/EXTERN symbols, relocation, and object module linking
- 8 character labels with sorted symbol table
- Built-in symbol table cross-reference
- Nested *INCLUDES (source file chaining/nesting)
- Lower case support (Model 3 only)
- Over 175 pages of documentation including Z-80 Technical Manual
- Full Screen Text Editing and much, much more!

M-ZAL provides a modular, structured development environment that makes programming your TRS-80 a truly enjoyable experience. Order yours today. (Requires 32k, 2 disk system)

#1050-10	Mod 1	$149.00
#1250-10	Mod 3	$149.00

"We've been using M-ZAL heavily for over two months now, and it's been worth its weight in gold."
—Chuck Tesler
PROSOFT

"...well written software that is long overdue."
—Bruce Douglass
80 Microcomputing
BASIC Editor Review

"During the first 3 days of use, M-ZAL saved more money in programmers time than we had paid for the product."
—William Denman
Author of Asylum
MED SYSTEMS

"The editor has at least doubled my productive capacity...Having spent a good deal of money on so called 'utilities' that don't work, it's a delight to find one that not only lives up to its claims but surpasses the advertising."
—BASIC Editor user

Our choice for the best TRS-80 DOS:

DOSPLUS 3.4	Mod 1	$149.00
DOSPLUS 3.4	Mod 3	$149.00

CAU Development Software— The Professionals Choice:

Full Screen Text Editor for BASIC: we were the first to give the TRS-80 this indispensable ability! A must for every BASIC programmer:

#1210-20	Mod 1 & 3	$29.95

T-ZAL: custom designed for the Model 3, this tape based assembler has many of M-ZAL's most desired features.

#1250-20	Mod 3 only	$49.95

XBUG: for machine language analysis and debugging—this tiny (2.5k) but powerful program lets you do it all!

#1020-10	Mod 1	$19.95
#1220-10	Mod 3	$19.95

FILEXFER: transfer any disk file between two TRS-80s. Use direct cable link or RS modems. Mod 1 and Mod 3 can be mixed. RS-232 and disk reqd.

#1040-10	Mod 1 and 3	$49.95

Call or write for more information.
All products require level II.
Mail orders specify catalog #.
Send check, money order, or MC/VISA numbers and expiration date to:

Computer Applications Unlimited
P.O. Box 214, Dept. ABM Rye, New York 10580
(914) 937 6286

N.Y./ State residents add applicable sales tax.
Outside USA and Canada add $10.00 shipping.
Dealer Inquiries Invited.

MasterCard VISA

COMPUTER APPLICATIONS UNLIMITED™
a div. of CAU, Inc.

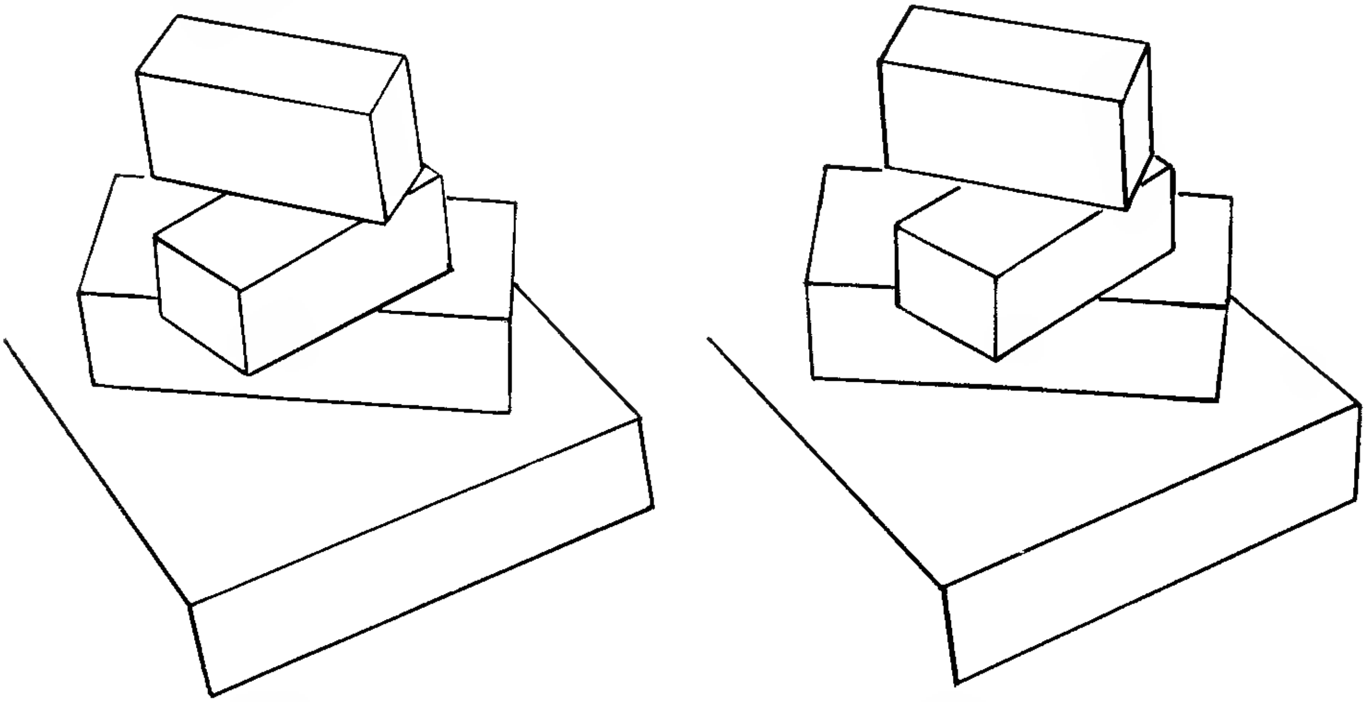
✓120

TRS-80 is a TM of Tandy Corp. M-ZAL is a TM of CAU, Inc. Z-80 is a TM of Zilog, Inc.

isting image. I used the last method.

Taking the Photo

Ideally, 3-D photographs are taken using a stereoptical camera—two-lensed animals that allow "real-time" photography of moving objects. However, if things are stationary, you need only a single camera. Take the photo, then carefully move the camera a few inches to your left or right. The distance you move the camera depends on the intensity of depth you would like—the



Figs. 2a and 2b. A line drawing of the two images. These drawings were done on tracing paper so X-Y coordinates could be derived for computer input.

farther, the deeper, up to the point where your eyes can no longer resolve the separate images into a single one. Start with about three inches, and move up to a foot—the closer the object, the smaller the move. Take the second photo using the same film, focus, depth of field, exposure time, and f-stop, and be sure to keep the same vertical position; a tripod helps. Otherwise, the pictures you take will make sense only if you hang your head to one side. Figure 1 shows the setup, and Photos 2a and 2b are a stereoptical result with a six-inch spacing.

"To view this pair of photographs in 3-D... focus on the images while relaxing your eyes, but this time crossing them."

To view this pair of photographs in 3-D, place the magazine at a comfortable distance, focus on the images while relaxing your eyes, but this time crossing them. Again, it is essential to relax,

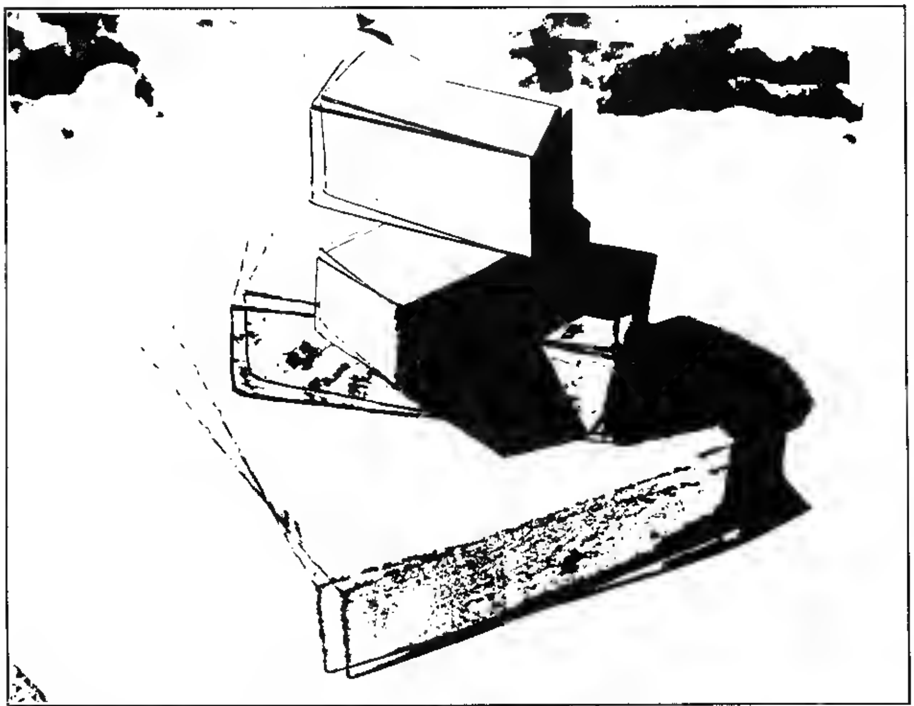
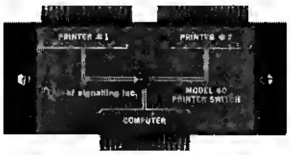


Photo 4. Overlay composite of two views of the blocks, exposed on Kodalith sheets from the same film used for Photo 3.



or else you will see only one of the two images. You want your eyes to "float" together, so each sees a separate but very similar image. The result is an extremely effective simulated 3-D.

From a stereoptical photo, a composite and simplified result must be developed to use as computer input. I made the pile of wooden blocks as the easiest first endeavor because of its straight lines and sharp angles. Photos 3a and 3b are the same pile of blocks as in Photos 1a and 1b but photographed using high-contrast document film and reproduced on similar high-contrast paper. If your original objects have been well detailed (I blackened the edges of the original blocks as you can see in Photo 1), then this is an easy way of obtaining a pair of clear line drawings of the original objects—especially if, like me, you do not have a good drawing hand.

Next, I produced an aligned overlay of the two black-and-white photos. In order to work up a composite 3-D image in two colors, I would need some sort of guide, and the aligned overlay would do the job. Each photo was again printed, this time on 8 by 10 sheet film. These were aligned and taped together for a composite. Photo 4 is the result, and is ready for use as computer input. However, I wanted to have a sneak preview.

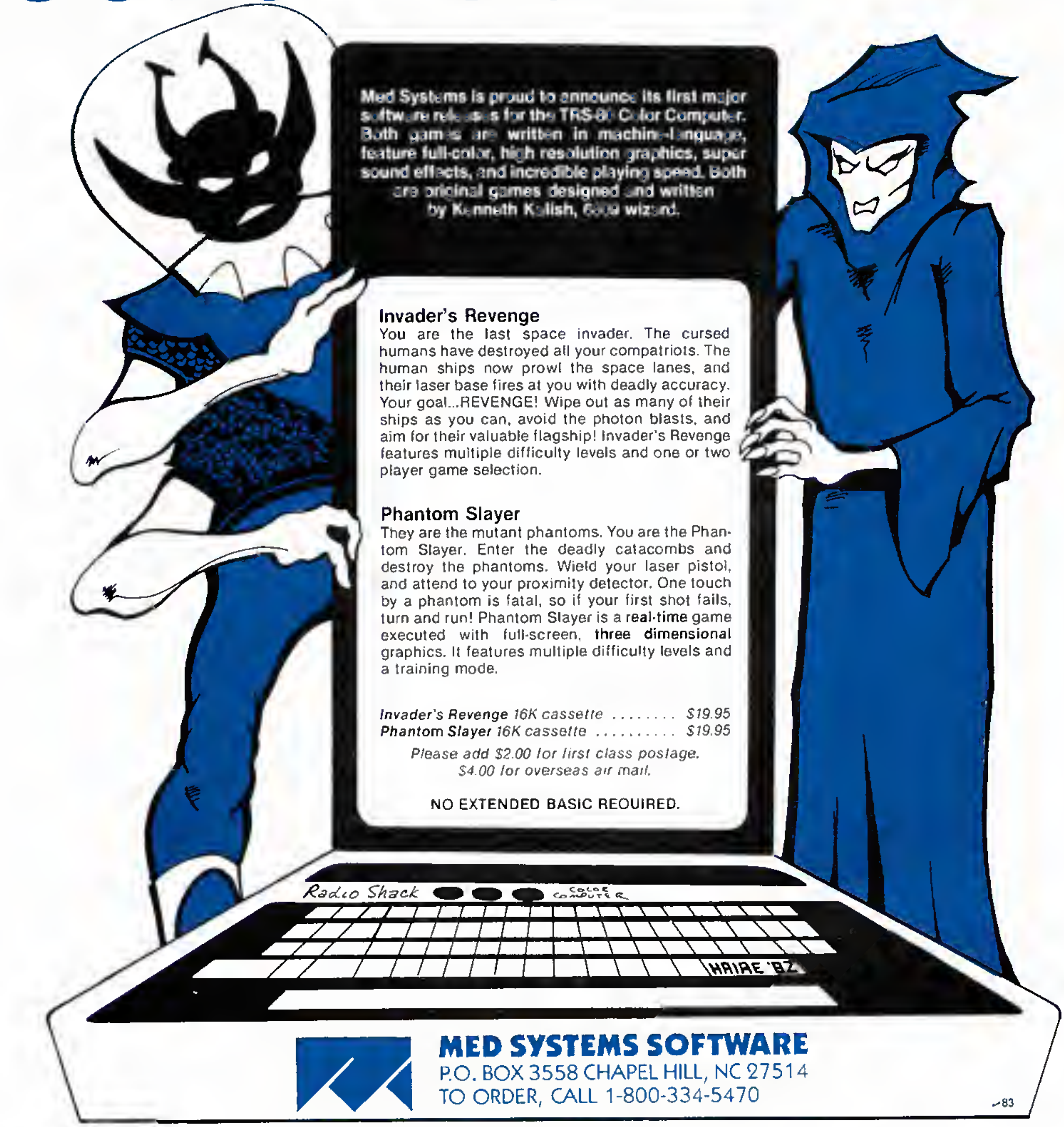
Color Composite

The color composite photo was tricky, if only because I had to do the thing in darkness. I taped the film overlay to an exposure frame, and focused the negative on its matching image. I added magenta Cibachrome filters (a total of .90 filtering), inserted the color photo paper, lifted aside the negative overlays, and made an exposure. I

```
10 PCLEAR8
20 CLS
30 REM * UPPER IMAGE OVERLAY
40 DATA 35,6,77,18,77,40,69,51
50 DATA 69,29,77,18,69,29,26,18
60 DATA 35,6,26,18,26,36,69,51
70 DATA 65,49,41,61,26,48,40,42
80 DATA 26,48,26,62,40,76,40,61
90 DATA 40,76,81,58,81,42,77,40
100 DATA 81,42,74,47,81,42,81,47
110 DATA 95,49,88,70,86,88,10,74
120 DATA 10,57,26,60,10,57,10,74
130 DATA 86,88,88,70,64,66,999
140 REM * LOWER IMAGE OVERLAY
150 DATA 24,19,30,9,74,18,74,38
160 DATA 69,50,69,29,74,18,69,29
170 DATA 24,19,24,38,64,49,44,60
180 DATA 25,50,38,43,25,50,25,65
190 DATA 42,77,42,60,42,77,80,60
200 DATA 80,42,76,40,80,42,72,46
210 DATA 80,42,80,46,91,48,84,86
220 DATA 90,68,60,65,90,68,84,86
230 DATA 8,78,8,60,25,62,999
240 PMODE3,1:SCREEN1,1:PCLS:SCREEN1,1
250 READA,B,C,D
260 LINE (A*2,B*2)-(C*2,D*2),PSET
270 READE:IFE=999THEN290ELSEREADF
280 LINE -(E*2,F*2),PSET:GOTO270
290 COLOR7,5
300 READ A,B,C,D
310 LINE (A*2,B*2)-(C*2,D*2),PSET
320 READ E:IFE=999THEN340ELSEREADF
330 LINE-(E*2,F*2),PSET:GOTO320
340 GOTO340
```

Program Listing 1

NEW! FOR THE COLOR COMPUTER!



Or see your dealer.

See List of Advertisers on Page 563

placed a black sheet over the color photo paper, flipped back the film overlay, matched the second negative to its image, removed the magenta filtering, and added an equal intensity of cyan (blue). Finally, I flipped back the negative and removed the cover sheet, and the exposure was made.

The result was a color composite that can be viewed with 3-D glasses. It is Photo 5.

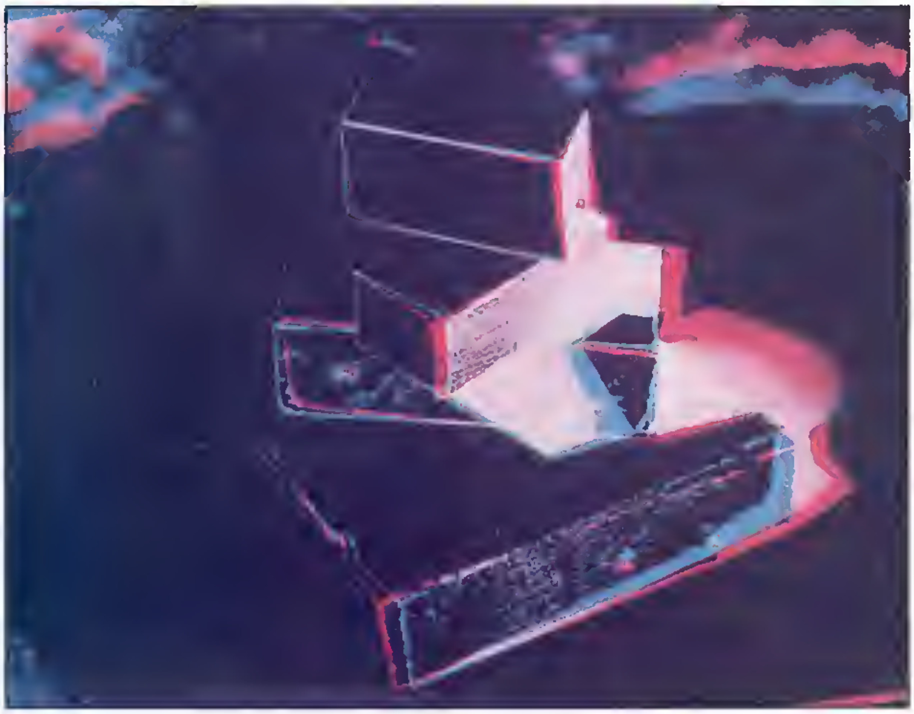


Photo 5. Color composite of two views of the blocks, exposed on Cibachrome-A paper alternately with .90 magenta and .90 cyan filters, using the overlay in Photo 4 as a guide for the double exposure. The white streaks in the center are the shadows, since this is a negative color print. Use your 3-D glasses to view this photograph.

Into the Computer

I had several ideas for this process, including an ill-fated one that involved extending one of the joysticks with a portable radio antenna and "tracing"

". . . an imaginative contraption with lots of pulleys and strings, made out of an Erector Set, fell to rubble on the first use."

the image into the computer. Unfortunately, the accuracy just wasn't there. X-Y tablets were too expensive, and an imaginative contraption with lots of pulleys and strings, made out of an Erector Set, fell to rubble on the first use.

All that remained was manual input. Fortunately, Extended Basic's flexible Line command could be used. I prepared a tracing-paper version of the two images, placing them on graph paper, and marking down the X-Y coordinates. These were fed into the computer using data statements (see the Program Listing), and the image was drawn. A similar Basic program can be used with the LNW-80 for higher-resolution color imagery.

I'll leave the rotation and manipulation to those with more perceptive minds than mine; in the meantime, I trust this method will help readers get started with three-dimensional computer imagery. ■

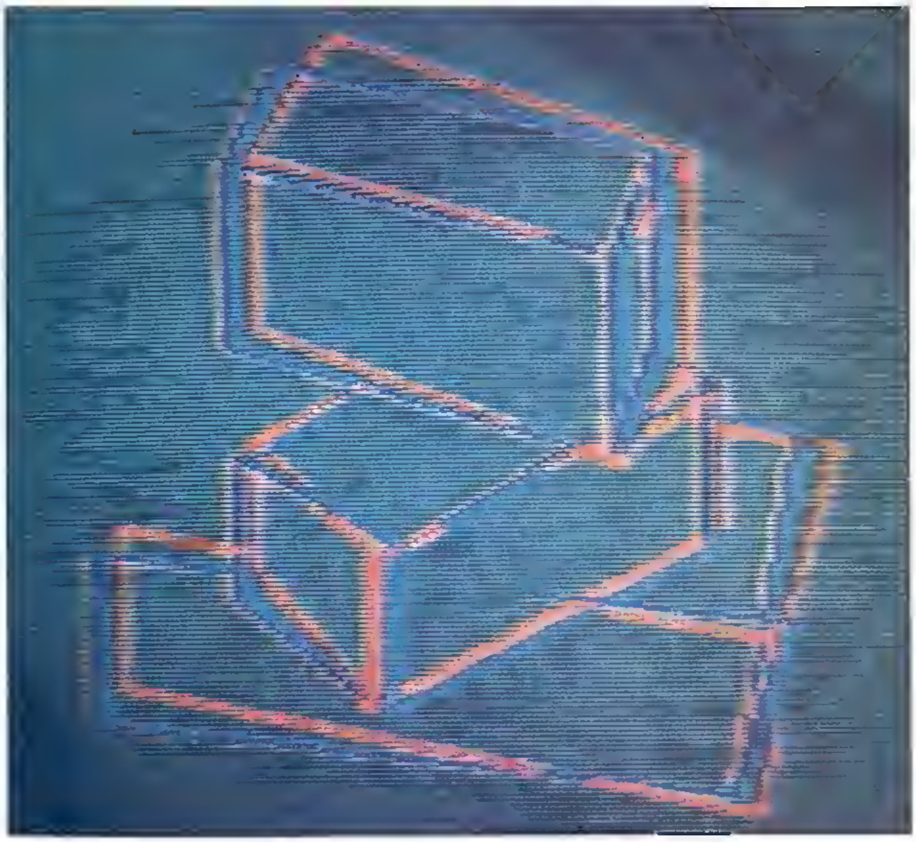


Photo 6. Screen photo of the results, using X-Y coordinates derived from the color composite in Photo 5. Unessential information ("noise") and shadows are left out of this result.

Dennis Kitsz may be reached c/o the Post Office, Roxbury, VT 05669.

Ed. note: The 3-D effect you get by running this program depends on the color and contrast adjustment on your monitor. Try to get as clear a picture as possible and match the colors of the 3-D glasses.

DOUBLE FEATURE
FOR YOUR TRS-80

HERE'S YOUR TICKET TO MANY THRILLING HOURS OF ENJOYMENT

TWO BRAND NEW GRAPHIC ADVENTURES IN ONE PACKAGE — FOR YOUR TRS-80

By Roger Schrag

SPOOK HOUSE

Horrors! A deranged madman has locked your unconscious form inside of a deserted creep house at a crazy carnival. You awake to some sobering facts: a bomb is set to explode, and only 30 real-time minutes stand between you and eternity. Somewhere, somehow you must locate and defuse the explosive — but we'd be less than honest if we thought you had more than a ghost of a chance! Brrrr — a real cliffhanger!

TOXIC DUMPSITE

Something's gone very, very wrong at the Toxic Dumpsite where the treatment and burial of deadly contaminants take place. The entire plant will explode like the Fourth of July in less than 30 minutes — unless you can avoid the many traps and protection systems and shut the plant down in time. Of course, time is the one luxury you haven't got. . . !

This is it — the blockbusting debut of the long-awaited Graphic Adventures! Now, your TRS-80 can hobnob with the best of 'em. Over 50 full screens of sharp, quality graphics breathe a realism and vitality into Adventuring, the likes of which you've yet to see on the TRS-80. Both Graphic Adventures feature game save, extensive vocabulary, real-time action (yes — we said real-time!) and accept FULL sentences! Exclusive area showings brought to you by Adventure International. Now playing at a TRS-80 near you!

TRS-80 48K Models 1 & 3 disk 012-0164 — $39.95

To order, see your local dealer. If he does not have the program, then call 1-800-327-7172 (orders only please) or write for our free catalog.

Adventure INTERNATIONAL

PRICES SUBJECT TO CHANGE

Published by ADVENTURE INTERNATIONAL
a subsidiary of Scott Adams, Inc.
BOX 3435 • LONGWOOD, FL 32750 • (305) [illegible]

Something-Or-Other 3-D

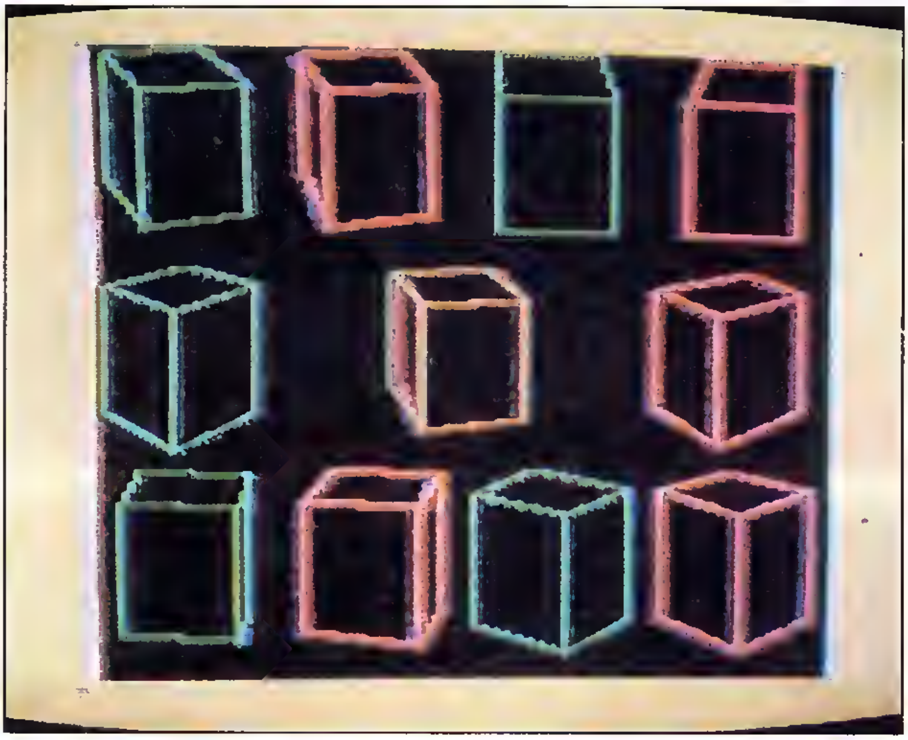
by Jake Commander

With great perseverence, Jake Commander has come up with this stereoscopic 3-D program that features a rotating cube in two colors.

Why is it I always walk into these things? Fools rush in where angels fear to tread, so I'm told, which makes me a 24-carat sucker.

"I've got a great idea," says Eric Maloney, our managing editor, after a sudden brainstorm. "We'll do an anniversary issue featuring 3-D graphics and give away glasses to our readers. You'll be able to write something for it, won't you, Jake?" The temptation was too much for me, I'm afraid. I instantly promised a something-or-other that would fit the bill. Little did I realize how the fates were aligning themselves to make the potentially simple something-or-other a whopping task.

I thought about the concept for a while to see if any good ideas surfaced. Nothing. I was convinced I could do something in a manner similar to some magazine pictures I'd seen as a child. The pictures were of countryside scenery and all in startling 3-D—lots of trees on the horizon with cows' rumps jutting out menacingly close to the eyeballs.



Jake's 3-D Program as It Appears on the Screen

The method employed was to print the left eye's view in green and the right eye's view in red. By placing a red filter over the left eye, the red image disappeared, thus leaving the correct image behind. Using a green filter over the right eye, the other side was accounted for. With two separate images entering both eyes, the brain fell hook, line, and sinker for the appearance of three dimensions. It was intriguing how convincing the illusion of depth could be, considering the simple separation of two images from a flat piece of paper. Similar methods are, of course, used in cinematography to produce those scary movies.

I felt there just had to be a way to achieve the same illusion using a two-color separation on the Color Computer. I was reminded of a program we'd already published, the idea of which was to present a three-dimensional point of view from any perspective in space. The clever part of the program was the way it decided what was in front of or behind each object from a set of geometric shapes that appeared in true perspective. That gave me the idea at last: How about a simple geometric shape such as a cube rotating before your eyes and seeming to do so in 3-D? Good enough, but how on earth was I going to simulate those overlapping colors I'd seen in that magazine?

At first I toyed with the idea of actually placing the images side by side using two colors in a medium-resolution graphics mode. I had to be kidding; the separation needed would have been sufficient to require a frown and deeply crossed eyes in order to make both images overlap.

Finally, I resorted to the good old

animators' technique using image retention. If I alternated a red and green image fast enough, they'd actually appear to be in the same place and no crossed eyes would be called for. As I had to use multiple images anyway to make the cube rotate, it was simply a matter of alternately switching between the two colors—one for each eye. I craftily chose a clockwise direction that ensured that the rotation naturally followed the images seen by the eyes in a left-right, left-right fashion. This helps the flow of movement so it appears to flow smoothly even without the glasses. Not only that, but after 90 degrees of rotation, the cube would be back in its original position and I could repeat the procedure ad infinitum.

I knew I could get the color separation by using the color artifacts I'd gotten many times while experimenting with color graphics in PMODE 4. By setting odd or even dots, it's possible to draw figures in two distinct colors. The colors are blue and red (or green and red, depending on your television). As long as the 3-D glasses could filter the colors to get two separate images, the illusion seemed feasible. I did a couple simple experiments drawing lines from Basic and was convinced that the separation was sufficient. Armed with a lot of theory, an experiment or two, and the idea coalescing in my mind, I had to complete the work in England—an event that didn't help one bit.

The first thing to be done was to decide how many angular views of the cube would be needed to give a smooth flow. The sequence of images would also have to be stored somewhere in memory, and it seemed reasonable to merely draw them onto the screen and leave them stored there. As long as there was a destination on the screen for the actual rotation, there would be no need to remove any image in the animation sequence. By dividing the screen into a 3-by-4 grid, I could get 12 cubes—or more accurately, six left/right images. That didn't leave room for the moving picture itself, so I dropped one out to allow five double images. Remembering I needed only 90 degrees of rotation, an image every 18 degrees or so would produce a steadily changing picture.

At this point, I should have been able to use an algorithm from somewhere in my software library to generate perfectly aligned cubes at every 18 degrees. The problem was my library was now 3,000 miles the wrong side of the Atlantic and this project was becoming more and more urgent. The only thing for it was to do it by hand.

So I took a Rubik's cube, placed it on a piece of paper marked at the required angles, and drew 10 pictures onto a piece of graph paper. First I had to close my right eye and draw, keeping my head in a predetermined position relative to the cube, and then the same thing with the left eye—for the total of five angles. Once drawn on graph paper, I calculated the X and Y coordinates for every point at each corner of every cube.

By feeding these coordinates to the Color Computer and writing an algorithm to join them together with lines of the right color, the drawings would be on the screen ready to animate. This explains those data statements in lines 6000–6900; each one represents a different cube.

By this time, it was dawning on me what I'd let myself in for. Not only was my software library 3,000 miles away from me, but so was my Color Computer and reference material on the 6809 microprocessor, which I needed to help write a small machine-code subroutine. This routine was needed to copy the successive images into place fast enough to fool the eye into seeing smooth action.

I was beginning to think I'd set myself an impossible task. The only recourse was to beg, steal, or borrow the required items. At last, armed with a 6809 reference book, lots of pictures of cubes, lots of data on these pictures, and a rough draft of the program, I found myself in a Tandy computer store in front of their Color Computer. To add a little spice to the pressure I was under, they even placed a young programmer next to me "to watch and learn." Can you imagine those odds?

Naturally, the program didn't work the first time. Nor the second. Nor the third or fourth. I lost count. After the second day, the cubes were being drawn nicely into position on the screen. If nothing else, the young pro-

```
100 PCLEAR4:PMODE4,1:PCLS:A$=STRING$(103,32):AD=VARPTR(A$):AD=PE
EK(AD+2)*256+PEEK(AD+3):DEFUSR=AD
120 FORX=0TO103:READOP:POKEAD+X,OP:NEXT
130 SCREEN1,1
1000 READXS:IFXS=-1THENGOSUB2000:GOTO1000ELSEIFXS=-2THEN2500ELSE
READYS,XE,YE
1050 XA=XE-XS:YA=YE-YS
1100 IFYS>YE THENI0=-1ELSEI0=1
1200 IFXS>XE THENI1=-1ELSEI1=1
1400 IFABS(XA)<ABS(YA) THENIX=ABS(XA/YA)*I1:GOTO1600ELSEIFXA=0TH
ENIX=0ELSEIX=ABS(YA/XA)*I0
1500 GOSUB1700:PSET(XP,YP,1):IFXS=XE THEN1000ELSEXS=XS+I1:YS=YS+
IX:GOTO1500
1600 GOSUB1700:PSET(XP,YP,1):IFYS<>YE THENYS=YS+I0:XS=XS+IX:GOTO
1600ELSE1000
1700 IFLR=1THENXP=XS OR1 ELSEXP=XS AND254
1900 XP=XP+(XO AND255):YP=YS+YO:RETURN
2000 IFLR=1THENLR=0 ELSELR=1
2100 XO=XO+64:IFXO=320THENXO=448
2200 IFXO=256ORXO=512THENYO=YO+64
2300 RETURN
2500 X=USR(0)
5000 DATA158,186,95,52,4,52,16,141,48,53,16,53,4,92,193,10,39,23
8,52,4,193,4,39,17,193,5,39,8,193,6,39,17,48,8,32,225,48,136,24,
32,220,220,186,139,8,31,1,32,212,220,186,139,16,31,1,32,204
5100 DATA220,186,139,8,203,12,31,3,16,142,0,64,52,32,16,142,0,4,
236,129,237,193,49,63,38,248,48,136,24,51,200,24,53,32,49,63,38,
230,16,142,12,0,49,63,38,252,57
6000 DATA12,0,12,12,12,0,46,0,46,0,52,12,52,12,12,12,12,12,12,52
,12,52,52,52,52,52,52,12,-1
6100 DATA12,12,18,0,18,0,52,0,52,0,52,12,52,12,12,12,12,12,12,52
,12,52,52,52,52,52,52,12,-1
6200 DATA22,0,12,10,22,0,56,2,56,2,48,12,48,12,12,10,12,10,12,48
,12,48,48,52,48,52,48,12,48,52,56,36,56,36,56,2,-1
6300 DATA26,0,10,10,26,0,58,2,58,2,46,12,46,12,10,10,10,10,10,48
,10,48,46,52,46,52,46,12,46,52,58,38,58,38,58,2,-1
6400 DATA30,0,10,6,30,0,62,6,62,6,40,14,40,14,10,6,10,6,10,44,10
,44,40,56,40,56,40,14,40,56,62,44,62,44,62,6,-1
6500 DATA32,0,8,6,32,0,60,6,60,6,36,14,36,14,8,6,8,6,8,46,8,46,3
6,58,36,58,36,14,36,58,60,46,60,46,60,6,-1
6600 DATA36,0,6,6,36,0,60,6,60,6,32,14,32,14,6,6,6,6,6,46,6,46,3
2,56,32,56,32,14,32,56,60,46,60,46,60,6,-1
6700 DATA40,0,6,6,40,0,60,8,60,8,28,14,28,14,6,6,6,6,6,46,6,46,2
8,56,28,56,28,14,28,56,60,48,60,48,60,8,-1
6800 DATA46,0,8,2,46,0,56,10,56,10,16,12,16,12,10,2,8,2,8,42,8,4
2,16,54,16,54,16,12,16,54,56,50,56,50,56,10,-1
6900 DATA48,0,10,2,48,0,54,10,54,10,14,12,14,12,10,2,10,2,10,40,
10,40,14,52,14,52,14,12,14,52,54,50,54,50,54,10,-2
```

Program Listing

grammer sitting next to me was learning the art of patience.

The final sting in the tail was that none of the cubes came out the right color. It was unbelievable. After all my hard work, applied theory, and having reached this far, surely the colors could at least come out right.

I checked the variables and they were correct; left cubes were being drawn at even-numbered pixel locations and right cubes at odd locations. Then I realized: I was working on a European PAL TV set—a different color standard than the NTSC All-American format I was used to. The color artifacts available on American TV sets were unobtainable in the UK. Thus, I was not going to be able to place my glasses on my nose and see the final fruits of my labors—a cruel trick of fate.

After convincing myself that the color problem was indeed the fault of the hardware and not the software, I debugged the machine code. The animation looked a real mess. The cubes rotated all right, but they danced and jiggled all over the place. Apparently, when drawing each cube, I'd moved my head this way and that. My drawing ability hadn't helped any either.

So I arrived at the last step, which was to analyze the sequence in slow motion and fine-tune each cube until the results were convincing. I now had something I could send the editors of *80 Micro* for a trial run to see if the colors were right and the illusion had succeeded.

Ten days later I received the cryptic message, "It works. Send the article ASAP." What, no medal?

There you have the whole tale including the evolution of the idea and solution of the problems using a brute-force approach.

The Technique

Five cubes are drawn in Basic using the algorithm in lines 1050–2300. Each cube is drawn twice, one for the left eye and the other at a slightly different angle for the right eye—a total of 10 cubes in all. Left cubes are drawn in blue (or green, depending on your monitor), and the right cubes in red. That's why simple Line commands can't be used—there's no control over the color artifacts.

Finally, a machine-code routine is entered at line 2500 that copies each image in sequence to the center of the screen at sufficient speed to achieve animation. The machine code is POKEd into a string from data statements at lines 5000–5100. It's written in position-independent code so it doesn't matter where it resides in RAM. Once the machine code is running, there's no provision to pause or stop it, so you'll have to use the reset button (the program will still be intact).

To set the thing up and get the best illusion, let the program draw two full cubes, then pause it with the shift @. Put your 3-D glasses on with the green filter over your left eye and adjust the monitor controls so that the left eye sees the maximum brightness of blue (or green) cubes and minimum brightness of red cubes. Now close your left eye and adjust so that you get maximum rejection of blue (or green) images and maximum brightness of red.

By doing two or three adjustments of color, brightness, and contrast, it's possible to get a fairly good separation of left and right images with the glasses supplied. Each cube should be about the same brightness balanced with maximum left/right separation to get the best effect. Let the program continue and watch the pretty cube. By the time you read this, I might even be able to enjoy it myself. ■

DISKovery Unlimited is bringing you the best in microcomputer systems, drives, peripherals, and software for TRS-80 and other popular brands. This month, take advantage of special system and disk drive values like these:

Complete TRS-80 Model III microcomputer system with disk drives

Includes specially modified A.M. Electronics Model III with 48K RAM, two 40-track, double-density disk drives, A.M. Electronics controller board, DOS and manuals, complete and ready to run *$1,695.00*

Same system as above, including RS-232C interface and MX82F/T dot matrix printer for correspondence-quality output *$2,395.00*

(Prepaid shipping when order is prepaid)

A.M. Electronics "Multiplier" for TRS-80 Model I systems

Fed up with unreliable double-density disk drive operation? "Multiplier" lets you upgrade from single-density without doubling your problems. . . . . *$ 99.00*

"Multiplier" with NEWDOS 80 . . . . *$199.00*

A.M. Electronics Mini Winchester hard disk drive

Completely plug-compatible with your TRS-80 system. Includes controller board and DOS Plus 4.0i. Delivers 6.7 megabytes of data capacity (unformatted). Typically uses one amp less than other hard disk drives. Excellent for word-processing, database management, financial planning, and other data-intensive applications *$1595.00*

New! Two-inch, high-performance floppy disk drives

Now you can save space *and* money. These drives are only two-thirds the height of conventional drives, and provide extras like door-lock solenoid control, selectable write protect (protect with or without notch—your choice!) and disk change/activity indicators. Shipped complete and ready to run.

Single-sided, 40-track 2″ drive w/case & power supply. *$199.95*

Double-sided, 40-track 2″ drive w/case & power supply. *$249.95*

Single-sided, 40-track 2″ drive, packaged for TRS-80 Model I, including case, power supply, cable, TRSDOS & manual *$249.95*

DISKOVERY UNLIMITED

✓37

3366 Washtenaw Ave., Ann Arbor, MI 48104 (313) 973-2315

TM TRS-80 trademark of Tandy Corp. Shipping costs extra except where otherwise noted. Prices effective through the 15th of cover month.

Solitary Maneuvers

James Wood

Since Colorful Maneuvers was published in the November 1981 issue of *80 Micro*, I have received several letters requesting more games for the 4K Color Computer. Colorful Maneuvers involves two players each controlling the direction of an ever-growing line. The first player to run his line into itself, the other line, or the border is the loser.

Solitary Maneuvers is a follow-up game to Colorful Maneuvers. The computer is the opponent.

Solitary Maneuvers is for the Colorful Maneuvers player when he is without a friend. It's a challenge of one's reasoning ability versus the computer's ability to look ahead of its line and make perfect turns as it approaches a barrier.

Listing 1 is a set-graphics game. If it becomes boring try Listing 2. It is a POKE-graphics version of the same game. With POKE graphics being twice as wide and tall as set graphics, the game proceeds much more rapidly. The POKE game also allows your line, the computer's line, and the border to be different colors. ■

James Wood, a teacher at Atwood Hammond High School, can be reached at 424 N. Missouri, Atwood, IL 61913.

Program Listing 1

```
0 REM JAMES W. WOOD,424 N. MISSOURI,ATWOOD,IL,61913,DEC 1981
1 CLS:PRINT:PRINTTAB(12)"TRAP"
2 PRINT:PRINT" USE THE ARROWS TO MANEUVER"
3 PRINT"THE LINE ON THE RIGHT."
4 PRINT" THE COMPUTER CONTROLS THE"
5 PRINT"LINE ON THE LEFT."
6 PRINT"  YOU MUST NOT DIRECT YOUR":PRINT"LINE INTO A COLORED SE
CTION."
7 PRINT:PRINT"PRESS <ENTER> TO CONTINUE"
8 A$=INKEY$:IFA$=""THEN8
9 CLS:INPUT"NAME";G$
10 S=RND(4):CC=0
11 IFS=1THENW$=CHR$(8):R$="A":GOTO15
12 IFS=2THENW$=CHR$(9):R$="D":GOTO15
13 IFS=3THENW$=CHR$(94):R$="E":GOTO15
14 W$=CHR$(10):R$="C"
15 U=12:X=51:V=10+RND(10):Y=10+RND(10)
16 CLS0:Z=RND(8):FORQ=0TO63:SET(Q,0,Z):SET(Q,31,Z):NEXTQ:FORQ=0T
O31:SET(0,Q,Z):SET(63,Q,Z):NEXTQ
17 Q$=INKEY$:IFQ$=""THENQ$=W$
18 W$=Q$:CC=CC+1
19 IFQ$=CHR$(8)THENX=X-1:GOTO23
20 IFQ$=CHR$(9)THENX=X+1:GOTO23
21 IFQ$=CHR$(94)THENY=Y-1:GOTO23
22 IFQ$=CHR$(10)THENY=Y+1
23 IFPOINT(X,Y)<>0THEN44
24 SET(X,Y,Z)
25 IFR$="A"THENU=U-1:GOTO29
26 IFR$="D"THENU=U+1:GOTO29
27 IFR$="E"THENV=V-1:GOTO29
28 IFR$="C"THENV=V+1
29 IFPOINT(U,V)<>0THEN30ELSE42
30 IFR$="A"THENONRND(2)GOTO31,32ELSE33
31 IFPOINT(U+1,V-1)=0THENU=U+1:V=V-1:R$="E":GOTO42ELSEIFPOINT(U+
1,V+1)=0THENU=U+1:V=V+1:R$="C":GOTO42ELSEGOTO46
32 IFPOINT(U+1,V+1)=0THENU=U+1:V=V+1:R$="C":GOTO42ELSEIFPOINT(U+
1,V-1)=0THENU=U+1:V=V-1:R$="E":GOTO42ELSEGOTO46
33 IFR$="D"THENONRND(2)GOTO34,35ELSE36
34 IFPOINT(U-1,V-1)=0THENU=U-1:V=V-1:R$="E":GOTO42ELSEIFPOINT(U-
1,V+1)=0THENU=U-1:V=V+1:R$="C":GOTO42ELSEGOTO46
35 IFPOINT(U-1,V+1)=0THENU=U-1:V=V+1:R$="C":GOTO42ELSEIFPOINT(U-
1,V-1)=0THENU=U-1:V=V-1:R$="E":GOTO42ELSEGOTO46
36 IFR$="E"THENONRND(2)GOTO37,38ELSE39
```

Listing 1 continues

0–9	rules
10–14	determines initial direction of lines
15	starting position of lines
16	draws border
17	Inkey for direction to turn your line
19–22	turns your line
23	determines if you lose
24	extends your line
25–28	extends computer's line, but doesn't display it yet
29	determines if computer must turn line
30–41	determines which direction computer should turn
42	extends computer's line
43	loops to line 17 to see if player wants to change direction
44–48	messages for winning or losing

Table 1. Line Description

The Key Box

Color Computer
4K RAM

Listing 1 continued

```
37 IFPOINT(U+1,V+1)=ØTHENU=U+1:V=V+1:R$="D":GOTO42ELSEIFPOINT(U-
1,V+1)=ØTHENU=U-1:V=V+1:R$="A":GOTO42ELSEGOTO46
38 IFPOINT(U-1,V+1)=ØTHENU=U-1:V=V+1:R$="A":GOTO42ELSEIFPOINT(U+
1,V+1)=ØTHENU=U+1:V=V+1:R$="D":GOTO42ELSEGOTO46
39 IFR$="C"THENONRND(2)GOTO4Ø,41
4Ø IFPOINT(U+1,V-1)=ØTHENU=U+1:V=V-1:R$="D":GOTO42ELSEIFPOINT(U-
1,V-1)=ØTHENU=U-1:V=V-1:R$="A":GOTO42ELSEGOTO46
41 IFPOINT(U-1,V-1)=ØTHENU=U-1:V=V-1:R$="A":GOTO42ELSEIFPOINT(U+
1,V-1)=ØTHENU=U+1:V=V-1:R$="D":GOTO42ELSEGOTO46
42 SET(U,V,Z)
43 GOTO17
44 FORE=1TO3Ø:SOUND2ØØ,1:SET(X,Y,Z):RESET(X,Y):NEXTE
45 A=A+1:CLS:PRINT"COMPUTER WINS":GOTO48
46 FORE=1TO3Ø:SOUND1ØØ,1::SET(U,V,Z):RESET(U,V):NEXTE
47 B=B+1:CLS:PRINTG$;" WINS"
48 PRINT:PRINTCC;"MOVES":PRINT:PRINT"TOTAL WINS":PRINT"COMPUTER"
;A:PRINTG$;B
49 PRINT:PRINT"PRESS <ENTER> TO CONTINUE"
5Ø A$=INKEY$
51 A$=INKEY$:IFA$=""THEN51ELSE1Ø
```

```
1 'JAMES WOOD, 424 N. MISSOURI, ATWOOD, IL, 61913
2 CLS:PRINTTAB(1Ø)"POKETRAP":PRINT:PRINT"USE THE ARROWS TO MANEU
VER":PRINT"THE LINE ON THE RIGHT."
3 PRINT:PRINT"THE COMPUTER CONTROLS THE":PRINT"LINE ON THE LEFT.
":PRINT
4 PRINT"YOU MUST NOT DIRECT YOUR":PRINT"LINE ONTO A COLORED SECT
ION."
5 PRINT:PRINT"PRESS <ENTER> TO CONTINUE"
6 A$=INKEY$:IFA$=""THEN6
7 CLS:INPUT"NAME";G$
1Ø S=RND(4)
11 IFS=1THENW$=CHR$(8):R$="A":GOTO15
12 IFS=2THENW$=CHR$(9):R$="B":GOTO15
13 IFS=3THENW$=CHR$(94):R$="C"ELSEW$=CHR$(1Ø):R$="D"
15 U=1258:X=1268
16 CLSØ:FORQ=1Ø24TO1Ø55:POKEQ,255:POKEQ+48Ø,255:NEXTQ:FORQ=1Ø24T
O15Ø4STEP32:POKEQ,255:POKEQ+31,255:NEXTQ
17 Q$=INKEY$:IFQ$=""THENQ$=W$
18 W$=Q$
19 IFQ$=CHR$(8)THENX=X-1:GOTO23
2Ø IFQ$=CHR$(9)THENX=X+1:GOTO23
21 IFQ$=CHR$(94)THENX=X-32:GOTO23
22 IFQ$=CHR$(1Ø)THENX=X+32
23 IFPEEK(X)<>128THEN44
24 POKEX,191:ONASC(R$)-64GOTO25,26,27,28
25 U=U-32:GOTO29
26 U=U+1:GOTO29
27 U=U+32:GOTO29
28 U=U-1
29 IFPEEK(U)<>128THENONASC(R$)-64GOTO3Ø,33,36,39ELSE42
3Ø ONRND(2)GOTO31,32
31 IFPEEK(U+33)=128THENU=U+33:R$="B":GOTO42ELSEIFPEEK(U+31)=128T
HENU=U+31:R$="D":GOTO42ELSE46
32 IFPEEK(U+31)=128THENU=U+31:R$="D":GOTO42ELSEIFPEEK(U+33)=128T
HENU=U+33:R$="B":GOTO42ELSE46
33 ONRND(2)GOTO34,35
34 IFPEEK(U-33)=128THENU=U-33:R$="A":GOTO42ELSEIFPEEK(U+31)=128T
HENU=U+31:R$="C":GOTO42ELSE46
35 IFPEEK(U+31)=128THENU=U+31:R$="C":GOTO42ELSEIFPEEK(U-33)=128T
HENU=U-33:R$="A":GOTO42ELSE46
36 ONRND(2)GOTO37,38
37 IFPEEK(U-31)=128THENU=U-31:R$="B":GOTO42ELSEIFPEEK(U-33)=128T
HENU=U-33:R$="D":GOTO42ELSE46
38 IFPEEK(U-33)=128THENU=U-33:R$="D":GOTO42ELSEIFPEEK(U-31)=128T
HENU=U-31:R$="B":GOTO42ELSE46
39 ONRND(2)GOTO4Ø,41
4Ø IFPEEK(U-31)=128THENU=U-31:R$="A":GOTO42ELSEIFPEEK(U+33)=128T
HENU=U+33:R$="C":GOTO42ELSE46
41 IFPEEK(U+33)=128THENU=U+33:R$="C":GOTO42ELSEIFPEEK(U-31)=128T
HENU=U-31:R$="A":GOTO42ELSE46
42 POKEU,2Ø7:GOTO17
44 FORE=1TO2Ø:SOUND1Ø,1:POKEX,191:POKEX,128:NEXTE:A=A+1:CLS:PRIN
T"COMPUTER WINS":GOTO48
46 FORE=1TO2Ø:SOUND2ØØ,1:POKEU,2Ø7:POKEU,128:NEXTE:B=B+1:CLS:PRI
NTG$;" WINS"
48 PRINT:PRINT"TOTAL":PRINT"COMPUTER",A:PRINTG$,B
49 PRINT:PRINT"PRESS <ENTER> TO CONTINUE"
5Ø A$=INKEY$
51 A$=INKEY$:IFA$=""THEN51ELSE1Ø
```

Program Listing 2

Colorful Language Instruction

by Dr. Alan F. Lacy and David Gorden

The Color Computer can be a useful tool in language instruction. Here's a graphic depiction of how the mouth makes certain sounds.

The Color Computer, because of its graphics capabilities, holds great promise for future use in education. Unfortunately, the computer is most often employed in the elementary and secondary grades, and not at the college level.

To fill this gap we have developed a program for use on a 16K Extended Basic Color Computer to teach German on the college level. You can adapt the program to teach other foreign languages or to teach English to speakers of other languages.

The Program's Purpose

The program was inspired by a course I teach on applied German phonetics. The class is intended to improve the German pronunciation of our more advanced students and to eliminate some of their American accent. Our students have had about 20 years of practice hearing and speaking English, and perhaps two or three years speaking German. As with most naive speakers, they have no idea what their mouth is doing when they speak.

To help my students change their ingrained habits I spend some time describing the structure of the sound-producing organs of the mouth. While this has been a help in the past, it is static.

Now I use the Color Computer to dynamically show the different configurations the mouth can take on during the production of different vowels. I chose vowels rather than consonants because German consonants are almost identical with their English counterparts, while the vowels are noticeably different and difficult for an English speaker to master.

Photo 1

Using the Program

The program gives you two options: You can go directly to an abstract schematic drawing, including all the vowels in their correct relative position in the mouth, or you can specify a certain vowel.

The first option is useful for gaining the overall picture. This option is a subroutine that you can enter either here, before displaying any one vowel, or again between displays of the individual vowels for reorientation. If you don't use this subroutine the program will ask you to input a vowel. This can be any simple vowel listed (see Program Listing 2, line 20), or

> *The Key Box*
>
> **Color Computer**
> **16K RAM**
> **Extended Color Basic**

diphthong. You can also specify whether you want a nasal or non-nasal version of the vowel. After you have entered a vowel, the program draws the mouth and tongue and displays the completed drawing (Photo 1.)

If you specify a diphthong, the graphics screen is called at the first tongue position, then the shift of the tongue and lips is drawn visibly. With German vowels, the position of the lips is also important in vowel production. Therefore the program displays a schematic front view of the lips to show whether they are rounded or spread. In English the lips are always rounded for the vowels u and o, as in boot or bow (and arrow), and spread for i and e, as in heat or bet. German has vowels, the so-called Umlaut vowels, where the tongue is in the front of the mouth, as for English i or e, but the lips are rounded as for English u or o. The program shows this. During the display of the diphthong oi, the change of lip position is also shown.

At this point the display is held until you make further input. If you hit any key except enter the program will loop back to the question "Diagram? (Y/N)." This means that after each time a vowel is shown, you can return to the diagram showing the schematic containing all the vowels. If you press N the program will move on to the input statement again, and you can enter a new vowel for display.

Pressing the enter key during vowel display enables the pointer routine. A menu is displayed from which you can pick any of 13 predefined positions in the mouth, or define your own position. The routine returns immediately to the graphics screen, this time with an arrow pointing at the chosen spot. Hitting any key erases this arrow. Then you can call the pointer routine again or return to show the schematic.

Once the program is begun, the only way to end it is by using the break key. Inappropriate input at any other point goes to a default routine and recycles to a predetermined point.

Program Description

When you turn on the Color Computer, it automatically reserves four pages of video memory (4 by 1.5K bytes), and 200 bytes of string space. However, if these numbers are changed any time while the computer is on, either by direct command or as part of a program, the computer uses the new parameters until they are changed again. When we revised this program, we included line 5 as a safety measure. If the program does not load, you will have to issue the direct commands NEW: PCLEAR4:CLEAR50 to make sure there is enough free memory available.

Another solution is to write a separate loading program, as we did when using five pages of video memory (see Program Listing 1.) This assures that there will be no problem in relocating the main program if a different PCLEAR command is given. This also makes it possible to eliminate line 5 from the main program. If you choose this method (highly recommended whenever your program requires more than four pages of video memory), it is best to enter this program, then CSAVE it under an appropriate name. Issue the New, the appropriate PCLEAR and the Clear commands directly, then key in the main program. CSAVE the main program directly after the loader program, but without a name. From this point on you will experience no problems in loading and running your program. The repetition of line numbers in the two programs is no problem—when you CLOAD the main program it overwrites the old program.

Line 10 dimensions the two arrays used for the arrow, and for returning the original graphics after the arrow is erased.

Line 50 clears and provides a blue

text screen. A blue screen is much less glaring when switching back and forth between text and graphics with its black background. It also avoids a momentary flash of garbage on the screen that the CLS0 command gives every time you call the text screen.

Lines 60–70 set the graphics mode 4, the highest resolution, requiring four pages of video memory. They also set the starting page at page 1, and clear all this memory, making it ready for drawing. Lines 100–130 draw a small arrow in the lower left corner of the screen and fill it in (Paint). The information for this drawing is then stored in the array AR, with full graphic detail (G) for later use. Finally, the drawing is erased using a Preset and BF (box fill) command. If this were not done, the arrow would always be present in the lower left corner when the graphics screen was called. Incidentally, you can use this technique in any program when you want to erase only a certain portion of the screen.

Lines 140–280 draw the lower portion of the head from the level of the lower teeth on down. Up until this point, nothing has been displayed on the screen. If you would like to see the computer draw, insert a line 90 SCREEN1,1. This calls the graphics screen, and you can see the arrow being drawn, erased, and then the lower mouth appear. This portion stays fixed. In PMODE4, each page of video memory contains the information for 1/4 of the graphics screen. Even though the first drawing is done at the middle of the screen, it is stored in pages 3 and 4 of video memory.

Lines 450–460 represent the reentry point after each vowel display, giving the possibility of going to the schematic. Line 300 skips this option the first time through, since it had already been given earlier (line 15). Here and at other places in the program we used an INKEY$ routine (lines 8000–8010) to perform two functions. It holds the present display until some input is given and it is used as the check for processing commands. This approach eliminates the use of the INKEY$ subroutine followed by an input command, which requires another key-

Program Listing 1

```
5 CLEAR 50
10 DIMAR(16,12),AE(16,12)
15 CLS:PRINT"   FOR  SCHEMATIC, PRESS  <S>,     OTHERWISE HIT AN
Y OTHER KEY     TO  CONTINUE":GOSUB 8000
16 IF I$=CHR$(83)THEN GOSUB 10000
20 CLS:PRINT"AVAILABLE OPTIONS ARE:":PRINT"A A: E E: I I: O O: U
 U:":PRINT"AI AU OI":PRINT"UMLAUTS: UE U:E OE O:E":PRINT"SCHWA:
@":PRINT"TYPE 'N' AFTER ANY VOWEL TO SHOWNASALIZATION.":PRINT:PR
INT"PRESS ANY KEY FOR NEW VOWEL."
30 PRINT"TO ACTIVATE THE POINTER,  PRESS <ENTER> DURING THE VOWE
L DISPLAY. TO EXIT, PRESS ANY KEY TWICE  EXCEPT <ENTER>."
40 PRINT:PRINT"PRESS ANY KEY  WHEN READY.":GOSUB8000
50 CLS3
60 PMODE4,1
70 PCLS
95 'DRAW FIGURE
100 LINE(12,176)-(4,182),PSET:LINE-(12,188),PSET:LINE-(12,184),P
SET:LINE-(20,184),PSET:LINE-(20,180),PSET:LINE-(12,180),PSET:LIN
E-(12,176),PSET
110 PAINT(11,179) 'DRAW ARROW
120 GET(4,176)-(20,188),AR,G
130 LINE(4,176)-(20,188),PRESET,BF 'ERASE ARROW
140 CIRCLE(112,75),80,,.4,.17,.3 'MOUTH BOTTOM
150 LINE(88,106)-(86,96),PSET 'LOWER TEETH #1
160 LINE-(83,108),PSET 'LOWER TEETH #2
170 CIRCLE(86,96),18,,.9,.28,.45 'LOWER LIP #1
180 CIRCLE(65,106),10,,.8,.4,.85 'LOWER LIP #2
190 CIRCLE(60,117),8,,.9,.75,1 'LOWER LIP #3
200 CIRCLE(75,127),15,,.87,.22,.68 'CHIN
210 LINE (75,139)-(120,134),PSET'LOWER JAW
220 CIRCLE(118,159),25,,1,.77,1 'ADAMS APPLE
230 LINE(143,160)-(144,181),PSET 'NECK
240 CIRCLE(210,101),35,,2.5,.4,.54 'WINDPIPE
250 LINE(157,101)-(155,123),PSET 'FRONT WINDPIPE #1
260 LINE-(163,120),PSET 'FRONT WINDPIPE #2
270 CIRCLE(62,151),105,,1.5,.97,.04 'GLOTTIS
280 LINE(181,156)-(179,181),PSET 'BOTTOM WINDPIPE
300 GOTO500
450 CLS3:PRINT"DIAGRAM? (Y/N)":GOSUB8000
460 IF I$=CHR$(89)THENGOSUB10000
500 GL=0:Y=15:A=0
510 CLS3:LINE INPUT"WHAT VOWEL? ";V1$
520 IF RIGHT$(V1$,1)="N" THEN A=1:V1$=LEFT$(V1$,LEN(V1$)-1)
530 IF V1$="A:"THENY=11
540 IF V1$="U:" OR V1$="I:" OR V1$="U:E" THEN Y=18
600 CIRCLE(120,82),50,,.55,.15,.35 'BASIC TONGUE #1
610 CIRCLE(120,106),30,,.6,.5,1 'BASIC TONGUE #2
620 PAINT(125,95) 'BASIC TONGUE #3
1000 LINE(233,181)-(233,15+Y),PSET 'BACK OF NECK
1010 IFA<>1THEN CIRCLE(77,Y+35),30,,.6,.08,.2:CIRCLE(130,Y+75),5
0,,.8,.65,.93:GOTO1030 'ROOF
1020 CIRCLE(77,Y+35),30,,.6,.08,.2:CIRCLE(130,Y+74),50,,.8,.65,.
7:CIRCLE(117,85+Y),60,,.8,.72,.92 'ROOF NASAL
1030 CIRCLE(131,Y+71),50,,.9,.52,.58 'TOP TEETH #1
1040 LINE(81,Y+70)-(76,Y+61),PSET 'TOP TEETH #2
1050 CIRCLE(65,Y+60),8,,1.5,1,.59 'TOP LIP
1060 PRESET(65,Y+72) 'ERASE
1070 CIRCLE(41,Y+49),20,,.6,.71,.1 'NOSE BOTTOM
1080 CIRCLE(36,Y+29),20,,.5,.28,.63 'NOSE
1090 LINE(22,Y+22)-(47,Y-11),PSET 'UPPER NOSE
1100 CIRCLE(95,71+Y),85,,.93,.58,.968 'UPPER NASAL
1110 IFA<>1THEN CIRCLE(95,81+Y),90,,.7,.64,.93:GOTO1130 'LOWER N
ASAL
```

Listing 1 continues

WHY PAY MORE FOR LESS WHEN YOU CAN HAVE BOTH 5" & 8" FLOPPIES FOR ONLY $169.95*

Some manufacturers charge you an arm and a leg for their Model-III# disk controller and then it only works with 5" floppy drives.

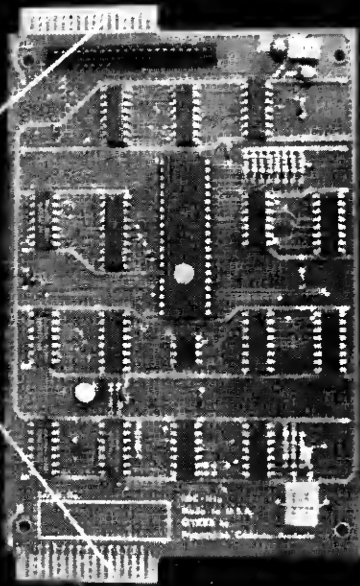
Some manufacturers sell controllers with cheaply made PC boards. These boards have poorly plated holes that can sometimes become intermittent. None are UL approved. Check our features and compare.

Almost none of the competitive boards have gold plated card edge connectors for reliability.

Some manufacturers sell controllers with part of the power supply included on the PC board to generate excessive heat to keep your disk controller warm on cold (and hot) days.

Some manufacturers get creative with their data separator circuitry to save money. These data separators generally work only with single sided disk drives.

Some manufacturers put only one connector on their PC board and give you a non-detachable umbilical cord hanging out the bottom to constantly get in your way.



FDC-3C FEATURES

☐ Gold Plated Card Edge Connectors for Reliability ☐ Proprietary Digital Phase Locked Lock Loop for NO MAINTENANCE operation with DOUBLE SIDED DRIVES ☐ Fully compatible with the Model-III# computer system ☐ Temperature Compensated Data Separator Circuitry ☐ Aerospace grade PC board material ☐ PC board material is UL Approved (R2 194V-0) ☐ Controls 5¼" Single and Double Density Drives ☐ Controls 8" Single and Double** Density Drives ☐ Controls Double Sided Drives ☐ 180 Day Warranty ☐ Switching Power Supply & Installation Kit available at extra cost.

Partial Dealer Listing (*denotes distributor)

*B.T. Enterprises (NY)	(516) 981-8568	Milo Systems (Mich)	(517) 487-1500
Compukit (Houston)	(713) 474-7342	Micro Systems Software (Florida)	(800) 327-8721
*Data-Mail (LA)	(213) 993-4801	Pioneer Electronics (Oregon)	(503) 668-8666
Freedom Technology (PA)	(215) 569-2381	Racet Computes (LA)	(714) 997-4950
Kif Computers (N. CA)	(707) 425-3472	Stevens Radio Shack (PA)	(800) 345-6279
*Level IV Products (Mich)	(313) 525-6200	Texas Computer Syst. (Dallas)	(800) 433-5184
Micro Data Supplies (Ohio)	(261) 951-6502	Traxx Computer (Chicago)	(800) 621-3102
Micro Facts (Florida)	(813) 876-4287		

Dealer Inquiries Appreciated **Attention OEMs and Manufacturers!!!**

Micro Mainframe can supply your component parts at attractive prices. We also have a series of switching power supplies available for both floppy and hard disk subsystems. Please call or write with your requirements.

MICRO MAINFRAME

11325 Sunrise Gold Circle Building E
Rancho Cordova, CA 95670
(916) 635-3997

#Model-III is a Tandy Trademark

*Installation Kit and Power Supply available at extra cost.
**Double Density 8" Operation requires 2 MHz. CPU speedup.

Listing 1 continued

```
1120 CIRCLE(95,81+Y),90,,.7,.64,.8:CIRCLE(110,73+Y),60,,.9,.79,.
96 'LOWER NASAL
1130 LINE(176,96)-(176,49+Y),PSET 'UP WINDPIPE
2000 GOSUB3000:IF V1$=""THEN2500
2010 SCREEN1,1
2020 GOSUB8000
2100 IFI$<>CHR$(13)THEN2300
2110 CLS:PRINT"SELECT POINTER POSITION:":PRINT:PRINT" A NOSE":PR
INT" B NASAL PASSAGE":PRINT" C UPPER LIPS":PRINT" D UPPER TEETH"
:PRINT" E ALVEOLAR RIDGE":PRINT" F SOFT PALATE":PRINT" G VELUM"
2120 PRINT" H LOWER TEETH":PRINT" I TONGUE":PRINT" J FRONTAL VIE
W, LIPS":PRINT" K GLOTTIS":PRINT" L VOCAL CORDS":PRINT" M BACK O
F NECK":PRINT" N MANUAL OPERATION";
2130 GOSUB 8000
2140 PT=ASC(I$)-64:IF PT>14ORPT<1THEN2110
2150 IFPT=1THEND1=40:D2=40
2160 IFPT=2THEND1=90:D2=10
2170 IFPT=3THEND1=74:D2=72
2180 IFPT=4THEND1=86:D2=72
2190 IFPT=5THEND1=105:D2=52
2200 IFPT=6THEND1=140:D2=42
2210 IFPT=7THEND1=173:D2=63
2220 IFPT=8THEND1=90:D2=92
2230 IFPT=9THEND1=105:D2=89
2240 IFPT=10THEND1=48:D2=114
2250 IFPT=11THEND1=168:D2=118
2260 IFPT=12THEND1=183:D2=165
2270 IFPT=13THEND1=237:D2=150
2280 IFPT=14 THEN CLS3:INPUT"SUPPLY X AND Y COORDINATES";D1,D2
2290 SCREEN1,1:GET(D1,D2)-(D1+16,D2+12),AE,G:PUT(D1,D2)-(D1+16,D
2+12),AR,PSET:GOSUB8000:PUT(D1,D2)-(D1+16,D2+12),AE,PSET:GOTO 20
20
2300 CLS3:PRINTCHR$(128);:LINE(5,104)-(45,136),PRESET,BF
2500 PMODE1,1:PCLS:PMODE4,1
2510 GOTO450
3000 IF V1$="A:" THEN RETURN
3010 IFV1$="AI"THEN SCREEN1,1:GL=1:V1$="E":GOSUB4000
3020 IF V1$="AU" THEN SCREEN1,1:V1$="O":GL=1:GOSUB4000
3030 IF V1$="OI" THEN CIRCLE(130,108),23,,1.2,.52,1:PAINT(130,88
):SCREEN 1,1:GOSUB5010:GL=1:V1$="E":GOSUB4000
3040 IF V1$="A" THEN CIRCLE (120,100),35,,.6,.5,1:PAINT (120,85)
:RETURN
3050 IFV1$="O:" THEN CIRCLE(130,100),23,,1.2,.52,1:PAINT(130,80)
:GOSUB5010:RETURN
3060 IFV1$="O" THEN CIRCLE(130,108),23,,1.2,.52,1:PAINT(120,88):
GOSUB4000:GOSUB5010:IF GL=1 THEN V1$="U" ELSE RETURN
3070 IF V1$="U:" THEN CIRCLE(130,90),23,,1.3,.55,1.1:CIRCLE(115,
90),18,,.6,.1,.88:PAINT(130,80):GOSUB 5010:RETURN
3080 IF V1$="U" THEN CIRCLE(130,100),23,,1.3,.62,1:CIRCLE(115,90
),18,,.6,.1,.88:PAINT(111,85):PAINT(130,80):GOSUB5010:RETURN
3090 IF V1$="@" THEN CIRCLE(120,100),36,,.69,.5,1:PAINT(120,80):
RETURN
3100 IF V1$="E" THEN CIRCLE(101,93),15,,.80:PAINT(95,90):GOSUB60
```

stroke followed by an enter.

Line 500 reclears the text screen, and initializes the flags for diphthong (glide) GL, and for nasalization A, as well as the variable Y, which determines the degree to which the mouth is open (15 is the neutral position).

After a slight delay, the input command appears on the display, and this input is assigned to V1$. A Line Input command has to be used in line 510, since the colon represents vowel length (the simple input command would ignore this). Line 520 is a check for nasalization. If the right-most element of the input string is N, the flag A is set to one, and the input string (V1$) is redefined to eliminate this N.

Lines 530 and 540 reset Y in the case of an open vowel (a:) or closed vowels (i: u:, u:e). Otherwise Y retains its neutral value of 15. Lines 600–620 paint the basic tongue position (A:), and lines 1000–1140 complete the drawing of the head and mouth.

The check in lines 1010 and 1110 is necessary because of the option for showing nasalization. Nasalized vowels require that the roof of the mouth and the bottom of the nasal passage be drawn in a different position, allowing the velum to appear open. If the vowel is not nasalized, then the velum is drawn closed (lines 1020 and 1120).

The subroutine in lines 3000–3170 gives the individual draw commands for the different tongue positions. Line 3180 is the default routine for inappropriate input. Redefining V1$ as empty enables us to skip the SCREEN1,1 command upon return (see line 2000).

Not counting the diphthongs, there are three different jaw positions, 18 different tongue configurations, two (or three) lip positions, and two positions for the velum, which causes nasalization.

Lines 3010–3030 identify the diphthongs. For any of these three, the SCREEN1,1 command is given immediately, instead of waiting to return from the subroutine to line 2010. This means that the first tongue position is shown immediately. We divided each diphthong into three positions: beginning, intermediate and final. After calling the graphics screen, the flag for a diphthong is set to one, V1$ is redefined for the intermediate position, and a short time loop is entered. Processing drops through until V1$ is found again, and this is then drawn over the existing position. Since GL is still one, V1$ is redefined, and the subroutine is run through until the final member is displayed, again drawn over the existing display. Since there are three different cases (ai, au, oi) where each time three separate vowels will be displayed, the order in subroutine 3000–3170 is important. The order must be (ai, oi, au) followed by a, o (u), e, i (u). The position of the other vowels is unimportant.

The extra Paint commands in lines 3080 and 3120 are needed to display diphthongs because over-drawing is done. Without them some portion of the tongue would remain unpainted. Finally, the two subroutines at line 5010 and 6010 give the frontal view of the lips. In the case of oi, both possibilities will be displayed, with the spread lips drawn over the rounded lips.

Once the program has drawn the tongue position, execution returns from this subroutine. If V1$ is empty, the program jumps to line 2500, then recycles; otherwise, the graphics screen is called. Issuing two SCREEN1,1 commands has no adverse effect when you input the diphthongs. This display remains until you input again. Hitting any key but enter causes a jump to line 2300. This clears the text screen, and calls it up blank, while the next commands are carried out. The first of these, Line . . . Preset, BF, erases the area where the frontal view of the lips was displayed.

Line 2500 shows how to erase the upper half of the display. Shift momentarily to the PMODE 1,1, which requires two pages (pages 1 and 2) of video memory. Then the PCLS command automatically erases the first two pages and you shift back to PMODE 4. This technique leaves the

✓See List of Advertisers on Page 563

top of the screen erased and ready for drawing and the bottom half untouched. Control returns to line 450 to give the option for the schematic, and then to reset variables and ask for more input. The video memory setup makes it possible to reenter here, rather than returning to line 50 and having the whole lower half of the screen redrawn each time.

If the time delay doesn't bother you, two solutions allow you to run the program with four pages of video memory. The first of these is to send control back to line 50, as mentioned above. Simply delete line 2500 and change line 2510 to read GOTO 50. Line 2510 must also be changed to . . . THEN 50. The other way is to use a Preset and BF command. In this case, change line 2500 to LINE(20,96)–(178,0), PRESET,BF. Line 2510 remains unchanged.

The pointer subroutine is in lines 2100–2290. While the program displays the graphics, hitting the enter key enables this routine. This routine displays a menu listing 13 different positions where you can place the arrow. This routine lets you point out one particular part of the vocal tract. The most likely places are pre-defined, but there is also the option for manual operation, where you can supply your own X and Y coordinates. Line 2140 converts the input to usable form, and lines 2150–2270 set the correct coordinates.

The arrow displayed comes from the array AR, which we loaded earlier in the program (lines 100–120). Before we can use the Put command, we must save the graphics contents of the area where the arrow is to appear. The AE (arrow erase) array accomplishes this. The graphics information contained in the area given by the coordinates is first stored in AE, and then the arrow is displayed. Hitting any key puts the contents of AE back into the same area, thus returning what was there and erasing the arrow. The AR array is defined once at the beginning, and never changed, while the AE array is redefined every time you make a new choice. At this point a jump is made back to line 2020, where you can again call the pointer routine or return to the schematic input option.

Lines 10000–10230 are the subroutine for drawing the schematic of the mouth. This outline is shown on the text screen, so the entire graphics screen does not have to be erased and then redrawn after this subroutine. M$ and N$ in line 10000 define the characters used in lines 10140–10160 and lines 10170–10190 for drawing the lines separating the mouth into nine areas. L$ marks the short vowel with a comma, indicating laxness.

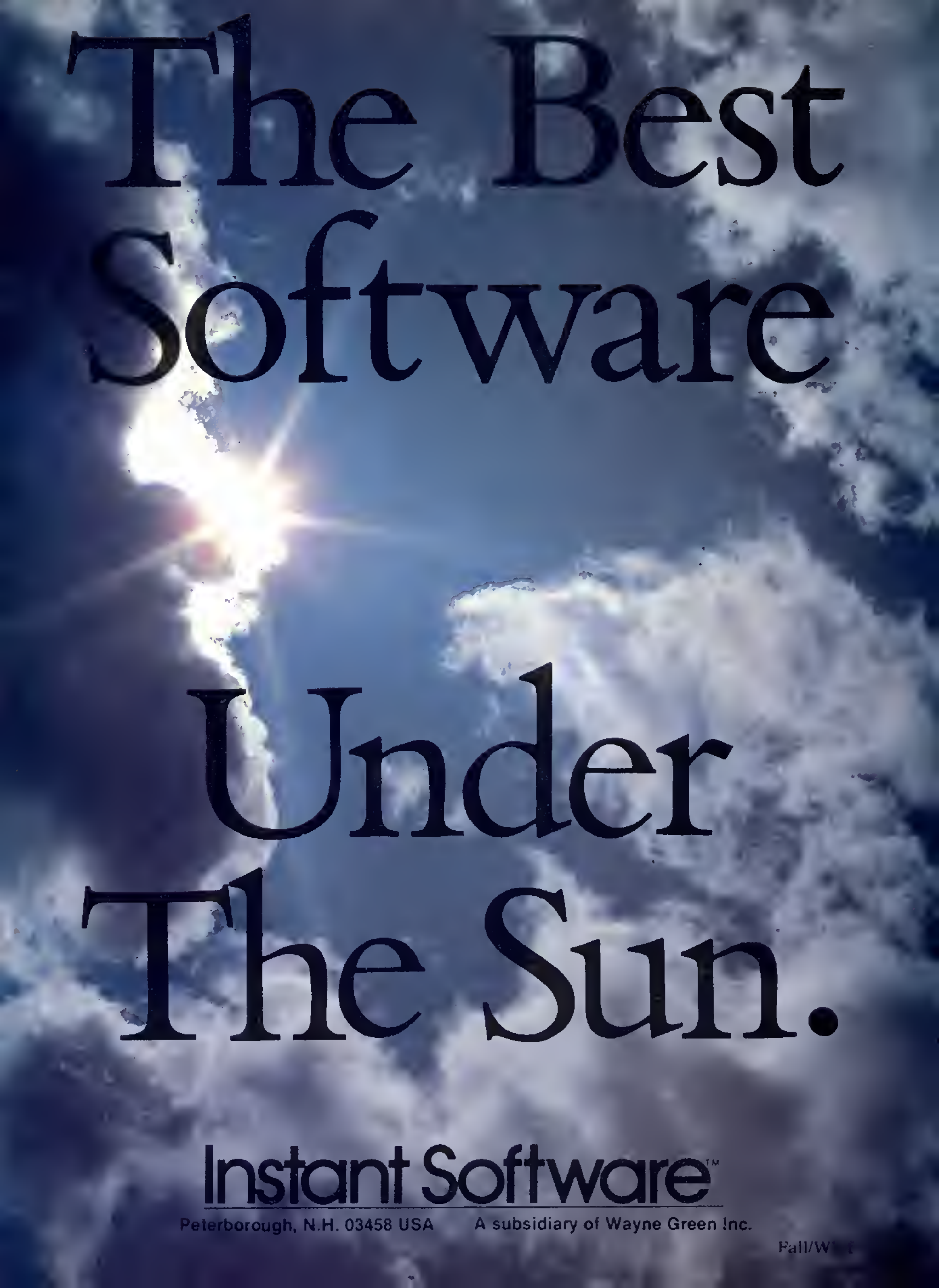
Lines 10010, 10030, 10060 and 10110 print the labels. These are in lowercase in the Program Listing because in the actual display they are reversed (green letters on a black background). To enter them this way, type the quotation marks, shift 0, the label, and then shift 0 again before closing the quotes. Line 10070, which reads in part PRINT AT 339, CHR$(93); CHR$ (91), prints left and right brackets in the lower right section of the central field. This marks the area of production for a German vowel, which is usually symbolized as an inverted V. This seemed the best solution, given the possible letters that could be used otherwise.

We would appreciate any comments readers might have for improvements to or other uses for the program. In return we are also willing to make cassette copies of this program available to anyone who will send us a blank cassette and the return postage. ■

Program Listing 2

```
10:GOSUB4000:IF GL=1 THEN V1$="I" ELSE RETURN
3110 IF V1$="E:" THEN CIRCLE(95,93),10,,1.3,.1,.83:CIRCLE(110,90
),17,,.6,.51,1.1:PAINT(90,90):GOSUB6010:RETURN
3120 IFV1$="I"THEN CIRCLE(100,90),15,,1.1,.2,.9:CIRCLE(125,91),2
3,,.49,.62,1.1:PAINT(90,85):PAINT(120,83):GOSUB6010:RETURN
3130 IF V1$="I:"THEN CIRCLE (111,87),18,,1.3,.39,.95:CIRCLE(128,
90),10,,1,.7,1.2:PAINT(111,75):GOSUB6010:RETURN
3140 IF V1$="U:E" THENCIRCLE(111,87),18,,1.3,.39,.95:CIRCLE(128,
90),10,,1,.7,1.2:PAINT(111,75):GOSUB 5010:RETURN
3150 IF V1$="UE" THEN CIRCLE(100,90),15,,1.1,.2,.9:CIRCLE(125,91
),23,,.49,.62,1.1:PAINT(90,90):GOSUB5010:RETURN
3160 IF V1$="O:E"THEN CIRCLE(95,93),10,,1.3,.1,.83:CIRCLE(110,90
),17,,.6,.51,1.1:PAINT(90,90):GOSUB 5010:RETURN
3170 IF V1$="OE"THEN CIRCLE(101,93),15,,.80:PAINT(95,90):GOSUB50
10:RETURN
3180 V1$="":CLS0:PRINT@195,"SORRY, YOU MADE AN ERROR.";:PRINT@23
0,"PLEASE BEGIN AGAIN.";
3190 FOR T=0TO800:NEXTT
3200 RETURN
4000 FOR T=0TO300:NEXT T
4010 RETURN
5000 'ROUND LIPS
5010 CIRCLE (25,120),20,,.85
5020 CIRCLE (25,120),15,,.85
5030 PAINT  (42,120)
5040 RETURN
6000 'SPREAD LIPS
6010 CIRCLE (25,120),15,,.4
6020 CIRCLE (25,120),20,,.4
6030 PAINT (42,122):PAINT(8,122)
6040 RETURN
8000 I$=INKEY$
8010 IFI$=""THEN8000ELSERETURN
10000 L$=",":M$=CHR$(128):N$=CHR$(140)
10005 CLS:PRINT"TO EXIT THIS ROUTINE, PRESS ANY KEY":FOR B=1TO2:
GOSUB4000:NEXT
10010 CLS:PRINT@6,"front";:PRINT@15,"central";:PRINT@26,"back";
10020 PRINT@70,"I: (U:E)";:PRINT@94,"U:";
10030 PRINT@96,"high";
10040 PRINT@134," I (UE)";:PRINT@156,"U";:PRINT@167,L$;:PRINT@18
8,L$;
10050 PRINT@263,"E:(O:E)";:PRINT@273,"@";:PRINT@283,"O:";
10060 PRINT@288,"mid";
10070 PRINT@328,"E (OE)";:PRINT@339,"[]";:PRINT@345,"O";
10080 PRINT@360,L$;:PRINT@377,L$;
10090 PRINT@36,STRING$(28,N$);
10100 PRINT@388,STRING$(28,N$);
10110 PRINT@433,"A";:PRINT@448,"low";:PRINT@465,L$;
10120 PRINT@497,"A:";
10130 PRINT@196,STRING$(28,N$);
10140 FOR P=78 TO 494 STEP 32
10150 PRINT@P,M$;
10160 NEXT
10170 FOR P=86 TO 503 STEP 32
10180 PRINT@P,M$;
10190 NEXT
10200 FOR P=68 TO 486 STEP 32
10210 PRINT@P,M$;
10220 NEXT
10230 GOSUB8000:RETURN
```

Dr. Alan Lacy (526 N. 14th St., Milwaukee, WI 53233) is an associate professor of German at Marquette University.


The Best Software Under The Sun.
Instant Software™
Peterborough, N.H. 03458 USA
A subsidiary of Wayne Green Inc.

161 PROGRAMS

☆ BUSINESS/APPLICATIONS
☆ UTILITIES
☆ EDUCATION
☆ GAMES

Index in Alphabetical Order

PLEASE NOTE:

- All programs specified for the TRS-80 Model I will also run on PMC and LNW computers.
- All TRS-80 Model I disk programs listed as compatible with the Model III must be converted by using the CONVERT command, as explained in your DOS Manual.
- All TRS-80 cassette programs are made to load from Radio Shack cassette recorders. Using any other recorders may require precise volume and tone settings.
- All TRS-80 disk programs are supplied with their own Disk Operating Systems.
- All Apple disk programs will boot on 3.2 AND 3.3 systems.
- All programs listed as having a "sound option" require the use of a speaker/amplifier or a cassette recorder with earphone.
- All Atari programs require the use of the Atari CXL 4002 BASIC computer language cartridge.

Cover photograph by Laurie MacMillan.
Group product photographs by Frank Cordelle.

MINI INDEX

OUR GUARANTEE

To Our Mail Order Customers

If you find this program to be defective within thirty days of the invoice date, you may return the program at no additional cost for exact replacement or full credit. You MUST enclose a copy of your invoice with the returned program.
If the program should become defective after thirty days, you may return any cassette with $4.00 or any disk with $5.00, along with a copy of your invoice to receive an exact replacement.

To Our Retail Customers

Because our retail dealers have various return policies, you must return any defective programs to the retail dealer it was purchased from.

Wordslinger

Put text-writing at your fingertips without disk drives and without a big investment! Designed for individual or small business use, the Wordslinger offers you complete text-creating and editing abilities, automatic formatting for letterheads and envelope addressing, and the ability to save your text on tape. Requires printer.

TRS-80 Tape Mod I and Mod III 16K 0129R *$29.95*

See page 4 for BUSINESS ANALYSIS, BOWLING LEAGUE SECRETARY and ELECTRONIC BREADBOARD. Page 5 has information on EASY CALC. See the inside back cover for the ad on our great GENERAL LEDGER package.

Find it Quick

Put an end to the Misplaced Information Syndrome with this disk-based personal librarian. A standard journal format records important magazine articles, memos, client lists, vendor advertisements, anything you need to keep track of. Subjects can be indexed or retrieved according to author, journal and date, or by up to nine keywords that you determine. Printer optional.

TRS-80 Disk Mod I and Mod III 32K 0258RD *$49.95*

See the back cover for our great GENERAL LEDGER PACKAGE ad.

MODEL ROCKET ANALYZER AND PREFLIGHT CHECK

Launch your TRS-80 into the high-soaring world of model rocketry! Two complimentary aeronautics programs: *MODEL ROCKET FLIGHT HISTORY PREDICTION* computes flight characteristics for almost any model rocket; includes data on engine and body tube for Estes, Centuri, Flight Systems, A.V.I., Astroport, C.M.R. and Kopter products; enter data for your own rocket designs! *WEATHER FORECASTER* predicts launch and flight conditions before lift off; you enter weather data and get an up-to-the-minute summary!

TRS-80 Tape Mod I & Mod III 16K 0024R *$24.95*

TRAFFIC ACCIDENT ANALYSIS AND RECONSTRUCTION

Nine programs (with multiple options) to solve the equations involved in motor vehicle accidents are included in this comprehensive, extensively documented North Star package. Reach accurate conclusions regarding the operational factors and dynamics contributing to highway accidents. Insurance companies, accident investigators, attorneys, fleet safety officials and police will all find that these programs make their accident investigations much more efficient and effective.

North Star Disk Horizon II and North Star DOS 5.0 48K 0173ND *$499.95*

Photo courtesy of Wang.

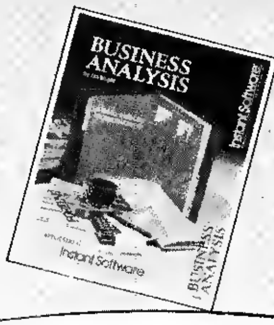
Scriptr

Scripsit users: Here's a universal printer driver that adds the features you need to your Scripsit package. With single letter commands, you can send any code to your printer from your text. Use all your printer's capabilities, use graphics, edit text as it goes to the printer, insert text to a specified place in the printout and never again lose your text because of a full disk! In addition, you can make your screen into a speed-reading trainer. Scriptr will control the Epson MX-80 and the Okidata Microline 80 printers and can be modified (for a nominal fee) to take advantage of all other printers that use the TRS-80 parallel port.

Requires Scripsit and a compatible printer.

TRS-80 Disk Model I only 32 and 48K 0380RD *$34.95*

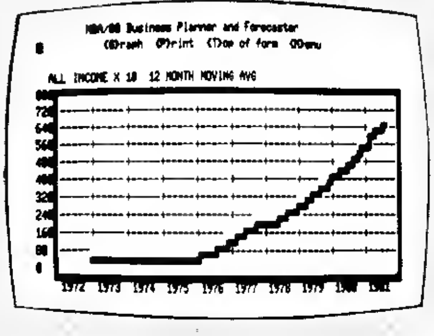
BUSINESS ANALYSIS



A flexible and professional time series analysis and forecasting instrument that allows you to analyze and appraise

- $$$ business climates and trends
- $$$ sales and product planning operations
- $$$ corporate stocks, trends and growth
- $$$ household budgets, accounts and energy

In fact, you can use BUSINESS ANALYSIS to examine and forecast ANY type of monthly, quarterly or annual data. You never exceed storage capacity; oldest data drops out leaving room for your most recent figures. A very flexible Editor program permits you to add, delete or modify at will. Forecast future data values using trend, moving average, seasonal or cyclic indices. Produce professional-looking charts, graphs and business reports in any of several output formats.

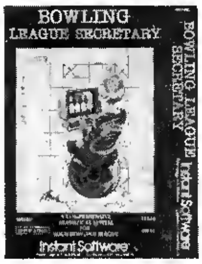


TRS-80 32K Tape Mod I & Mod III 0140R *$75.00*
TRS-80 32K Disk Mod I & Mod III 0152RD *$99.95*

the electronic breadboard

ELECTRONIC BREADBOARD assists in the design and analysis of analog circuits and can be used to evaluate voltages, currents, impedance, and the frequency response of any circuit as well as dozens of other applications. Students verify electrical theory by entering sample circuits and determining frequency response. The documentation includes a tutorial designed to increase the newcomer's understanding of electronics. An ideal program for audio component repair technicians, electrical engineers, electricians, and students of electronics.

TRS-80 Tape Mod I and Mod III 16K 0287R *$49.95*
TRS-80 Disk Mod I and Mod III 32K 0222RD *$59.95*
Apple Disk Applesoft 32K 0428AD *$59.95*



astrology ☆ ☆

This program, written by foremost astrologer/computerist Michael Erlewine, supplies the calculations and planetary placement data necessary for a trained astrologer to interpret. The horoscope cast will show planetary placement in the Zodiacal Circle and the twelve houses. Knowledge of Astrology is required to use this professional program.

TRS-80 Tape Mod I and Mod III 16K 0241R *$14.95*
Apple II Disk Applesoft 28K 0242AD *$19.95*

BOWLING

BOWLING LEAGUE SECRETARY

Here's a disk version of this essential program with all the above features PLUS...more storage capabilities (store up to 250 bowlers and 40 teams!), and the ability to record individual handicaps. You can utilize two different formats with the printout option, and, should you have any problems, simply type HELP to receive an explanation of what information is needed—complete with a sample entry.

TRS-80 Disk Mod I & Mod III 48K 0095RD *$49.95*

BOWLING LEAGUE STATISTICS SYSTEM

This program is in a league by itself! Transform your TRS-80 into a comprehensive record-keeping system that's ready to handle any bowling league. Simply enter in weekly scores and BLSS provides a complete list of bowler, team, and league statistics including information such as points won, team standings, high averages, games, and series. You can record information for up to 12 teams and 60 bowlers on a 16K system, and up to 16 teams and 90 bowlers with a 32K system. A variety of different scoring options will suit every league's needs. Printer optional.

TRS-80 Tape Mod I & Mod III 16K 0056R *$24.95*

Client Record/ Bill Preparation

Small to medium-sized businesses need the Client Record/Bill Preparation system. Create files which contain complete information about each client. Add charges for time, materials and expenses, and calculate and print detailed invoices. You may also search for and print client records, separate projects and charges.

Apple Disk Applesoft 32K 0284AD *$49.95*

PERSONAL BILL PAYING

Handle your household accounts like major businesses do—computerize them! Get a month-by-month listing of your current debts, bills already paid (w/check numbers!) and all other pertinent information necessary to professionalize your bill-paying methods!

TRS-80 Tape Mod I & Mod III 16K 0103R *$14.95*

ELECTRONICS DESIGN CALCULATOR

Dozens of complex calculations are computed and brought down to size with this comprehensive program. Calculations are given and displayed in their most readable form for reactance, resonant frequency, resistors and capacitors in parallel and series, along with over two dozen other complex formulas. Free yourself from the drudgery of endless calculating.

TRS-80 Tape Mod I and Mod III 16K 0204R *$14.95*

ELECTRONICS ENGINEER ASSISTANT

Hours of trial-and-error breadboarding are no longer necessary: your computer can be your lab, breadboard and test equipment. This package deals with problems in network analysis and microstrip design. Analyze cascaded combinations of up to 13 network elements. Design microstrip transmission lines for any medium where impedance is a critical factor. Flexible and efficient. This package is designed for both the professional and the hobbyist.

PET Tape Old and New ROM 8K 0085P *$14.95*
Apple Disk Applesoft 32K 0267AD *$19.95*

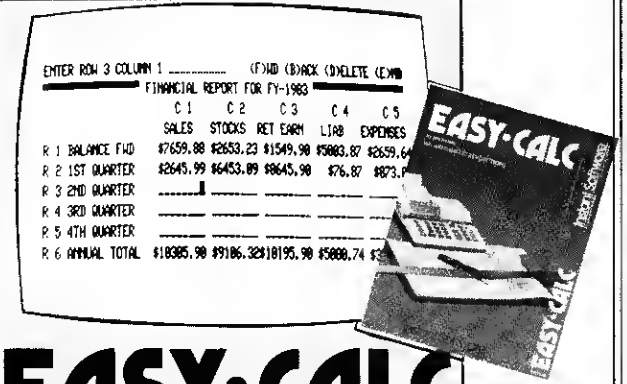
ELECTRIC >>

Master Plot

A professional graph plotting and printing package which includes these outstanding features:

*Data can be entered from the keyboard, from equations you enter, or from your BASIC programs.
*10 sets of data can be plotted on the same graph.
*Data is plotted using symbols you compose.
*You can choose the number of interpolated points between data points.
*The program automatically selects scale values for ease of interpretation.
*Graphs from 1" × 1" × 7" by 24" may be printed.
*The number of horizontal and vertical grids can be selected.
*A comprehensive manual will guide you step by step through all procedures.

Requires Epson MX-80[1] printer with the Graphtrax[2] modification.
TRS-80 Disk Model I only 48K 0435RD *$149.95*



EASY·CALC

Easy Calc will turn your TRS-80 into an electronic spreadsheet! Write your numeric data into rows and columns on your video screen. Then add, subtract, multiply, divide or exponentiate single values or complete rows and columns. Easy Calc contains a unique programmable calculator that allows you to enter and save an entire series of calculations that can be executed instantly by touching a single key. Additional features include: calculation of percentages, summation of all rows or columns, and labeling of all rows and columns. Easy Calc will handle up to 600 figures (e.g. 30 rows by 20 columns). Print your worksheet whenever you wish, or merge it with Scripsit word processor files for complete, professional reports.

TRS-80 Disk Model I only 48K 0269RD *$49.95*
TRS-80 Disk Model III 48K 0369RD *$49.95*

MASTER directory

Organize all your disks and disk files with this Master! Keep track of all your programs and files (up to 5,000!) on all your disks (up to 320!); locate programs even if you've forgotten their correct names; get master lists of all your data files—alphabetically, by data category, or extension!

TRS-80 Disk Mod I 32K 5005RD *$29.95*
TRS-80 Disk Mod III 32K 5004RD *$29.95*

WEATHER WATCH

Come rain or shine, you'll be "in the know" with this two-program weather package.

WEATHER FORECASTER Just enter past and present barometric pressures to obtain a weather "preview" for the next 24 hours.

WEATHER PLOT is an informative weather almanac that contains a "flood" of weather data. You have at your command records, charts, graphs, facts and figures on local weather for every major city in the U.S. and its possessions.

TRS-80 Disk Mod I and Mod III 32K 0316RD *$24.95*

HAM PACKAGE

Three programs make life easier for the amateur radio enthusiast... *BASIC ELECTRONIC FORMULAS*—Performs seven commonly used calculations including ohms law for DC circuits, voltage dividing/dropping, series resistance, parallel resistance, series capacitance, and R/C time constant.

DIPOLE ANTENNA—Designs a dipole antenna for any frequency.

YAGI ANTENNA—Designs a three-element Yagi antenna for any frequency within the ham bands.

Apple Disk Apple Soft 32K 0176AD *$19.95*
Pet Tape Old and New ROM 8K 0054P *$14.95*

QSL MANAGER

A must for ham radio operators. Make and review log entries, print summary reports of all entries, and search your entire file for specific information. Optional printing function.

TRS-80 Disk Mod I & Mod III 32K 0151RD *$24.95*

SOLAR ENERGY FOR THE HOME

For any homeowner seriously considering the advantages and economics of solar conversion, this package will provide practical guidance by computing a potential cost/benefits comparison between solar and conventional heating methods. *SOLAR ENERGY* even calculates the payback period to show if your contemplated investment will indeed save money.

Apple Disk Applesoft 32K 0235AD *$34.95*

NFL PROGNOSTICATOR

Keeps updated records of all NFL teams each season and gives accurate predictions of winners and point spreads each week! Just type in game scores each week and you receive screen displays and/or printouts of current league standings, team stats and next week's winners! Get the "inside line" BEFORE the weekend!

Apple Disk Applesoft 32K 0145AD **$24.95**

ENERGY audit

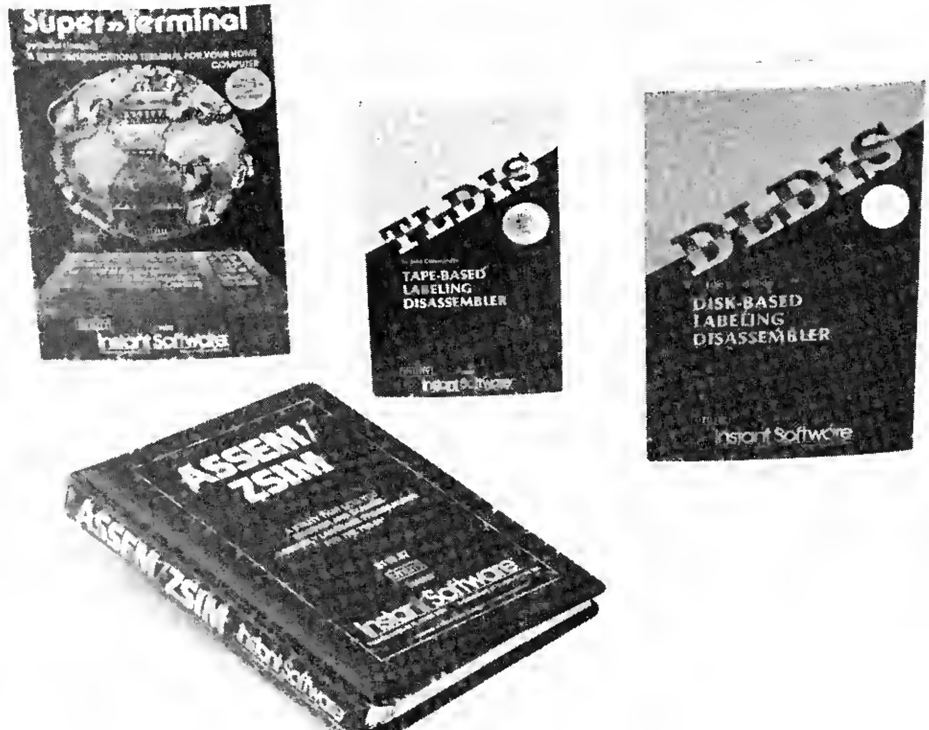
ENERGY AUDIT Save money and energy with this very comprehensive and easy-to-use package. Answer all your important home energy conservation questions on the dollars-and-sense level. Get a professional cost analysis BEFORE investing in expensive building materials. A favorite among energy consultants across the nation. A must for any conscientious carpenter, contractor, or home owner! (Disk version includes printer options)

TRS-80 Tape Mod I & Mod III 16K 0089R *$49.95*

TRS-80 Disk Mod I only 32K 0052RD *$75.00*

Photo courtesy of Data General.

UTILITIES



See page 7 for SUPER>>TERMINAL. Page 8 has information on TLDIS, DLDIS and ASSEM/ZSIM.

double listing

GRADE BOOK—0050R—**$14.95**—TRS-80 cassette—see EDUCATION listing
TEACHER—0065R—**$14.95**—TRS-80 cassette—see EDUCATION listing
OMNI CALCULATOR—0211RD—**$29.95**—TRS-80 disk—see EDUCATION listing
TEACHER'S AID—0214RD—**$39.95**—TRS-80 disk—see EDUCATION listing

COMPRESSION UTILITY PACK

Compact your BASIC programs into a smaller amount of memory—without sacrificing executable code! You get two super programming utilities that let you add all those little extras to your BASIC programs.

** *COMPRESS—80* fits in 265 bytes, deletes spaces and give your the choice of deleting or keeping REM statement line numbers in the program.

** *SUPERCOMPRESS* uses 767 bytes and packs your programs into the smallest possible number of multiple statement lines.

TRS-80 Tape 16K 0246R **$19.95**

DOSPLUS 3.4

The fastest, smoothest, easiest to use Disk operating system for your TRS-80! Jam-packed with powerful options: BASIC Array Sort, Tape/Disk—Disk/Tape Utility, Controlled Screen Input, Random Access and ASCII modification on Diskdump, BASIC Checks for Active "DO", Backup and Format from a "DO" file, and much, much more. Energize your operating system with DOSPLUS 3.4!

TRS-80 Mod I single density 5023R **$149.95**
TRS-80 Mod I double density 5024RD **$149.95**
TRS-80 Mod III 5025RD3 **$149.95**

RENUM/COMPRESS

Add these two powerful commands to your programming toolbox: RENUM allows you to expand program line number intervals and re-address as needed; and COMPRESS frees up memory space by reducing your program's space in RAM.

TRS-80 Tape Mod I only 16, 32, & 48K 0133R ***$14.95***

BASIC PROGRAMMING ASSISTANT

Write and debug your programs better with this marvelous programming tool! List all your program variables and ranges of variables; cross-reference GOTO & GOSUB line addresses; search your program listing for specific BASIC function words; print lines where any variable changes its value; and do much, much more—you're limited by your imagination only.

TRS-80 Tape Mod I Only 16, 32, & 48K 0203R ***$19.95***

ABE

Accelerate your writing and editing of Basic programs with the Advanced Basic Editor. Dramatically improves your Basic editing and programming capabilities by providing such indespensible features as Formatted Screen, Global Search and Replace, Line Copy Ability, Assignable Keys, Called From Basic. Advance your skills as a programmer with ABE.

TRS-80 Mod I 32K & 48K ***$19.95***
TRSDOS 2.3 or NEWDOS 80 v2 5026RD

KEY COMMANDER

Easy editing of programs and direct entry of graphics into Basic programs are yours with this highly sophisticated yet easy to use utility package. Features On-screen Editing, Graphic Printer Statement, Assignable Keys, and Total Flexibility. Automatically relocates into high memory and protects itself in memory. Enhance the power and capabilities of your keyboard!

TRS-80 Tape Mod I & Mod III 16K 5027R
$29.95

PROGRAMMER'S PRIMER

Learn and use the fundamental concepts of computer programming: decimal-to-HEX conversion; data storage; logic or program flow; sorting routines; and others. Presented in a pleasing and efficient format, sure to please any novice computer nut!

TRS-80 Tape Mod I & Mod III 16K 0245R ***$14.95***

ENHANCED BASIC

A comprehensive utility to add powerful new features to Level II BASIC: enhanced string handling capabilities, direct use of hexadecimal and octal constants, decimal to hexadecimal conversions, user defined functions in BASIC, the ability to call as many as 10 machine language subroutines from BASIC, a keyboard debounce routine, and the capability for remote keyboard operation from a cassette tape.

TRS-80 Tape Model I only 16K 0077R ***$24.95***

TRS-TESTS

The easy-to-use diagnostic utility that lets you check your ROM for bad bits, keyboard for defective keys, data and address lines for clean signals, and perform a series of RAM diagnostic checks. A must for anyone interested in nuts-and-bolts TRS-80 systems maintenance!

TRS-80 Tape Mod I Only 16K 0184R ***$14.95***

Disk Scope

Solve password mysteries, access riddles and relieve other disk-using frustrations! This easy-to-use package lets you locate files on disks, display track and sector in HEX and ASCII, and gain access to any file by constructing a suitable password. Works on single density disks only.
TRS-80 Disk Mod I Only 32K 139RD
$24.95

CASSETTE SCOPE

"Look" into your tapes! Locate lost files and filenames; find load addresses and entry points; read and display all addresses in HEX; do all of this quickly, easily and professionally!
TRS-80 Tape Mod I Only 16, 32, & 48K 0192R ***$14.95***

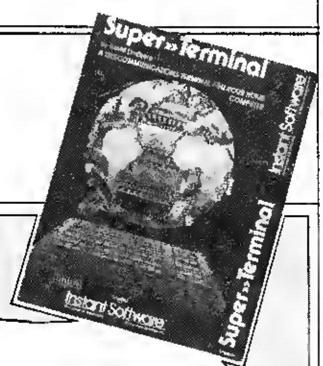
THE COMMUNICATOR

Provides an excellent, but economical means of communicating between HOST and REMOTE terminals. Lets you orchestrate the functions of your remote keyboard, RS-232C output and input ports, and your host terminal video display.
TRS-80 Tape Mod I only 16, 32 and 48K 0126R ***$14.95***

disk~tape exchanger

Manipulate and manage your magnetic media! Put your tape files on disk where they're easier to work with; put your disk files on tape for reasons of storage space and cost; manage your media with no mistakes!
TRS-80 Disk Mod I Only 32K 248RD
$24.95

Super»Terminal



Turn your ordinary TRS-80 into a Superterminal! Integrate all your microcomputer components with this super software. This package means SPEED: lets you simplify procedures for signing-on, transmitting files, and communicating with any remote computer system. It means POWER: integrate and orchestrate all communication pathways between your terminal and all peripherals. It means SIMPLICITY: comprehensive Menus permit you to select from many options at-a-glance. And it means VERSATILITY: make Special Command tables for arranging your various commands, make Control Key tables for adding previously inaccesible characters, and—it is compatible with any DOS! Other super features include direct XYZ cursor addressing and screen, cursor and printer controls, a text editor, HEX conversion utilities, plus compatibility with all present ROMs and DOSs! Get speed, power, simplicity and versatility—all in one super package! Requires RS-232C Interface.
TRS-80 Disk Mod I & Mod III 32K 5700RD ***$95.00***

In reference to SUPER>>TERMINAL (5700RD). . .

". . .we had tried several different telecommunications packages, all which either did not meet our needs, or were insufficient in user friendliness. When we received SUPER>>TERMINAL, it opened up an entirely new dimension to the things that we could do. Without your program, I fear that we would not have been able to manipulate information with the ease and confidence that we do now. Please continue to produce products of this quality."

Bruce A. Bergman
Micro Programmer/Analyst
Seattle Pacific University
Seattle, WA 98119

SUPER UTILITY PLUS

The most powerful program of its kind! A Machine language, stand-alone program with its own I/O routine. No ROM or DOS calls. Works on single and double density systems. Features ZAP, PURGE, FORMAT, DISK COPY, TAPE COPY, DISK REPAIR, MEMORY, FILE, and CONFIGURE SYSTEM functions. Very sophisticated; menu driven, single RAM resident program. A must for every serious TRS-80 disk installation.
TRS-80 Mod I & Mod III 48K 5022RD
$74.95

apple clinic

The programming aids package for the Apple programmer: DISK DOC aids in documenting and maintenance of Apple DOS 3.2; APDOC helps APPLESOFT II programmers; COMPARE documents the line-by-line difference between two versions of the same program.
Apple Disk Applesoft 32K 268AD
$29.95

DYNAMIC DEVICE DRIVERS

Reprogram your keyboard, video display and printer to make them perform better for you! Customize the commands your computer sends to its components and add some helpful features to your system!
TRS-80 Tape Mod I Only 16K 0228R ***$19.95***
TRS-80 Disk Mod I Only 16K 199RD ***$24.95***

*LABEL

Free yourself from the tyranny of confusing line numbers! Label your BASIC program subroutines: GOTO 9105 becomes GOTO BUBBLE SORT ROUTINE, for example. Write and debug your programs more efficiently!
TRS-80 Tape Mod I & Mod III 16K 0168R ***$24.95***

DISK EDITOR

Edit any track of any standard single-density disk in any drive: byte-by-byte! Now you can read and write to otherwise inaccessible sectors; modify system files; search for any eight-character string; and print any video display.
TRS-80 Disk Mod I Only 32K 180RD ***$39.95***

PROGRAMMER'S CONVERTER

Convert decimal numbers to binary, octal or hexadecimal—and back again. You get 1) straightforward conversions; 2) practice doing the conversions yourself; and 3) conversion quizzes. A great tool for beginning programmers!
TRS-80 Tape Mod I & Mod III 16K 0058R ***$14.95***

TERMINAL-80

Plug your TRS-80 terminal into the world! This package lets you use your computer from remote terminals, communicate with just about ANY other computer system via your modem, and even transfer program code over the phone! Requires Expansion Interface and RS-232 Serial Interface.

TRS-80 Tape Mod I & Mod III 16K 0130R ***$24.95***

ASSEM/ZSIM

Assembly language programmers: solve all your programming problems from Assem. to Z-sim! ASSEM, the 3-pass editor/assembler, uses little RAM: provides a powerful line editor, is compatible with any parallel-port printer: and executes w/out modification on 16, 32, or 48K systems, w/1, 2, 3 or 4 disk drives! And ZSIM, the machine code simulator/debugger, emulates instructions using simulated registers: displays mnemonics for each instruction using ASSEM's symbol table; works as well in ROM as in RAM! This package allows you to assemble directly to disk, tape or memory—directly from disk, tape or memory! Trace program execution through ROM to debug larger, more complicated programs with speed and accuracy!

TRS-80 Disk Mod I only 32K 365RD ***$119.97***

PET UTILITY I

Two great programming aids for the PET enthusiast! With MONITR you can edit, save and verify any machine-language program while still having access to BASIC; and use the PROGRAMMER'S CALCULATOR as a floating-point calculator and to do decimal-binary-octal-hexadecimal conversions!

PET Tape Old and New ROM 8K 0105R ***$14.95***

UTILITY II

Great programming aids formerly available only in more expensive utility packages: search through any Level II tape to find filespecs, merge any two BASIC programs or any BASIC program w/one or more machine-language programs!

TRS-80 Tape Mod I Only 16K 0076R ***$14.95***

TLDIS

The tape-based disassembler that automatically assigns labels to machine language program routines. You can send the disassembly to your printer or save it on tape for editing and reassembly using Radio Shack's EDTASM.

TRS-80 Tape Mod I & Mod III 16K 0230R ***$19.95***

ULTRA-MON

The unique and powerful machine-language monitor that displays, disassembles, traces (hardcopy trace disassembly, too!), modifies, relocates memory, prints and even relocates itself with simple commands. Designed for the beginner as well as the professional programmer.

TRS-80 Tape Mod I & Mod III 16K 5003R ***$24.95***

ZSIM

Debug larger, more complicated programs with this simulating, labelling debugger! Any debugger will enhance the usefulness of your assembler, but only ZSIM can make your programming tasks easier! ZSIM

*********RUNS* machine code instructions one-at-a-time at your bidding;

*****EMULATES* the instruction using simulated registers;

***INTERRUPTS* the simulations whenever any one of a large number of user-specified conditions are met;

*****DISPLAYS* mnemonics for each instruction, using convenient labels:

********LISTS* register contents and corresponding memory locations.

ZSIM works on ROM as well as RAM since you don't need breakpoints to retain execution control (although breakpoint operation is also available). This is the best tool available to examine your code in DETAIL!

TRS-80 Tape 16K 0376R ***$29.95***

DLDIS

The disk-based diassembler that automatically assigns labels to machine language program routines. You can send the disassembly to your printer to say it on disk for editing and re-assembly using Apparat's extension of EDTASM[1].

TRS-80 Disk Mod I Only 32K 231RD ***$24.95***

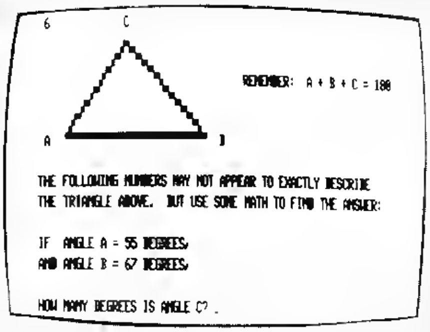
THE DISASSEMBLER

A single-pass, HEX-notation disassembler that sends output to tape or your line printer: shows displacement and absolute address of any relative jumps made by the disassembled program; displays any ASCII characters used in an LP or CP opcode.

TRS-80 Tape Mod I & Mod III 16K 0232R ***$14.95***

ed-u-ca-tion

Photo courtesy of Bell & Howell.



THE ELEMENTS

Here's a more interestng approach to learning the Periodic Table of Elements that every high school student will welcome. Quiz sections provide questions and answers that deal with element names, atomic numbers, weights and symbols. The reference section gives all information for each element.

TRS-80 Tape Mod I and Mod III 16K 0216R **$17.95**

TRIANGLE TRIG

Trigonometry made painless! Triangle Trig is a step-by-step guide to Triangular Geometry and Beginning Trigonometry that reviews all necessary mathematical skills. Abundant graphics, careful explanations, and practice using new skills in practical situations make this one of the best tutorial programs we've seen. Suitable for advanced upper-elementary students through adults.

TRS-80 Tape Mod I and Mod III 16K 0309R **$19.95**

WORDWATCH

Entertainment and education combined! Develop language skills using these four fun programs.

WORD RACE A game for two Grand Prix drivers who can define words quickly and accurately.

HIDE N SPELL Spot the misspelled word before you lose any points. Then spell it correctly!

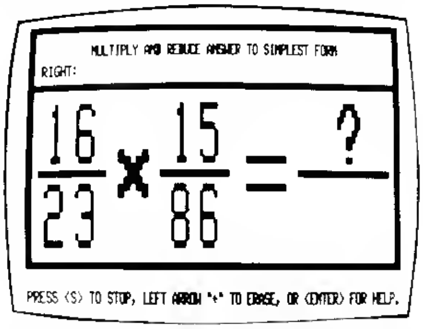
SPELLING BEE Spell the word you hear on a cassette recorder and the computer will give you hints.

SPELLING TUTOR The computer gives you hints: words spelled backwards, or with missing or mixed up letters. You spell the words correctly.

Teachers and parents can easily put their own words and definitions in all these programs. Even German vocabulary can be fun! Interest level—age 6 through 12.

TRS-80 Tape Mod I and Mod III 16K 0111R **$14.95**

Photo courtesy of IBM.



MATH MASTER

Practice makes perfect. And now students get a chance to practice math skills at their own pace and on their own. MathCard presents addition, subtraction, multiplication and division problems using the time-honored flash card method. MathFrac gives practice in adding, subtracting, multiplying and dividing fractions. 25 levels of difficulty are available for elementary through junior high school students.

TRS-80 Tape Mod I and Mod III 16K 0257R **$17.95**

In reference to TRIANGLE TRIG (0309R) . . .

"Excellent intro to Trig!!! Strongly suggest anyone who needs a review or who is planning to take Advanced Math take this course."

Laurence C. Lee
Kansas City MO 64138

POLYGONS

Lots of graphics and careful explanations introduce students to regular polygons. The program then delves into these related topics:

1. Circular approximation and calculation of pi.
2. Combinatorial associations of vertex points.
3. Tesselation of the plane or "mosaics" using regular and semiregular polygons.

Suggested grade level—high school through adult.

TRS-80 Tape Mod I and Mod III 16K 0244R **$14.95**

CAPITALIZATION

This educational program introduces and exercises the twelve rules of capitalization in English. Students can progress at their own pace. An optional Model Mastery worksheet is provided for student and teacher use. Recommended for ages 12 through adult.

Apple II Disk Applesoft 32K 0339AD **$24.95**

VIDEO SPEED-READING TRAINER

Increase your reading efficiency! If you read like t-h-i-s, Video Speed-Reading Trainer can train you to quickly recognize letters and numbers so that you'll begin to read whole phrases at a glance. Advance at your own pace to increased speed with better retention and recall.

For ages 12 to adult.

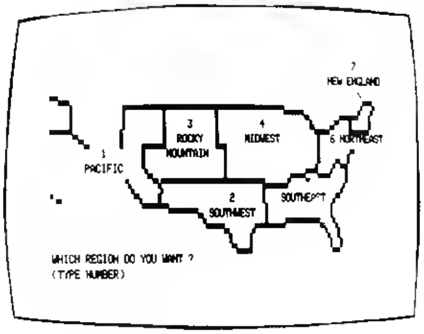
TRS-80 Tape Mod I and Mod III 16K 0100R **$14.95**

See page 10 for the GEOGRAPHY EXPLORER SERIES and page 11 for the TYPING TEACHER.

GEOGRAPHY EXPLORER: USA

MID EAST

Europe



GEOGRAPHY EXPLORER SERIES

Here are three programs that will turn your children's lessons into child's play. Essential geographical facts are presented in an entertaining manner with fun features like animated graphic rewards for correct answers. Practical features, such as lesson planning suggestions and study aids, assure the educational benefits of these superior programs. Disk versions provide an additional "Teacher Options" file that allows parents or teachers to choose question format, determine which facts a student can view, and select scoring methods. Upper elementary through adult.

GEOGRAPHY EXPLORER: USA contains three lesson sets that include every state's location, capital, area in square miles, and population as well as other pertinent information.

GEOGRAPHY EXPLORER: EUROPE and **GEOGRAPHY EXPLORER: MID EAST** provide three lesson sets that include each country's name, government, chief export, language, geographical features, and many other important facts.

GEOGRAPHY EXPLORER: USA

TRS-80 Tape Mod I and Mod III 32K 0086R $29.95
TRS-80 Disk Mod I and Mod III 32K 0071RD $49.95

GEOGRAPHY EXPLORER: EUROPE

TRS-80 Tape Mod I and Mod III 32K 0259R $29.95
TRS-80 Disk Mod I and Mod III 32K 0121RD $49.95

GEOGRAPHY EXPLORER: MID EAST

TRS-80 Disk Mod I and Mod III 32K 0276RD $49.95
TAPE NOT AVAILABLE

TEACHER

Create a computer teaching system for any subject! TEACHER allows you to record up to 20 questions and answers per lesson. Students can then do these lessons independently. You may choose to allow them to review the material beforehand, to receive hints, and to see animated displays as rewards for correct answers.

TRS-80 Tape Mod I and Mod III 16K 0065R *$14.95*

TEACHER'S AIDE

Teachers, parents and students themselves can record a sequence of questions with answers. Create your own lessons, quizzes, tests and learning games. Correct answers are rewarded by some very clever video animations. Revise/update your lessons at will. Save student score in separate limited access file!

TRS-80 Disk Mod I & Mod III 32K 214RD **$39.95**

GRADE BOOK

Use the speed and accuracy of your computer to calculate student grades. Grade Book enables you to do the following without hassles.

1. Calculate quarterly, monthly or other periodic averages using as many as five categories (tests, quizzes, homework classwork and projects).
2. Calculate semester and yearly averages.
3. Weight category and periodic scores, and convert raw scores to percentage points.

(No printer option.)

TRS-80 Tape Mod I and Mod III 16K 0050R **$14.95**

Photo courtesy of Bell & Howell.

OMNI CONVERTER

One liter of gas is how many gallons? Convert one unit of measurement to another within each of 10 categories; length, volume, mass, velocity, area, density, power, energy, pressure and stress, and temperature. With all possible units within each category (including metric), the number of conversions you can do instantly is almost infinite. You can even print your results! Optional printer.

TRS-80 Tape Mod I and Mod III 16K 0200R **$14.95**

ARCHIMEDES' APPRENTICE

Learn the formulas for determining the volume of various geometric figures and be quizzed on your new abilities. Shapes include parallelpipeds, prisms, pyramids, cylinders, cones and spheres. Suggested grade levels: 6 through 12.

TRS-80 Tape Mod I and Mod III 16K 0092R **$14.95**

SURVEYOR'S APPRENTICE

Learn the formulas for finding the areas within various geometric figures—AND how to use these formulas! This excellent learning tool even provides quizzes to sharpen your new skills to perfection!

TRS-80 Tape Mod I & Mod III 16K 0127R **$14.95**

OMNI CALCULATOR

Three simple-to-use programs that enable you to measure, calculate and convert quantities with ease:

ARCHIMEDES' APPRENTICE Learn how to find the areas of three dimensional figures. (See package #0092R.)

SURVEYOR'S APPRENTICE Learn how to find the areas of two dimensional figures. (See package #0127R.)

OMNICONVERTER Convert from one unit to another—effortlessly. (See package #0200R.)

TRS-80 Disk Mod I and Mod III 32K 0211RD **$29.95**

Beginner's RUSSIAN

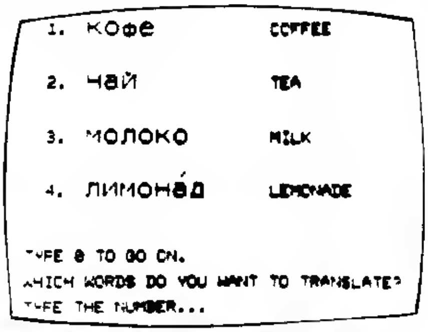
Learn Russian easily with this comprehensive language course. Learn to recognize the letters of the Cyrillic alphabet and practice pronouncing Russian words. As your skill increases, you'll advance to the next section, conquering the basics of Russian step-by-step.

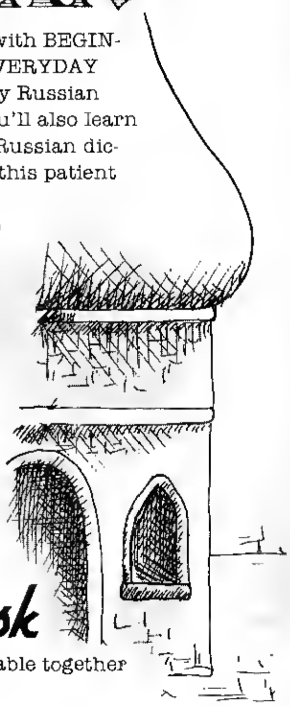
TRS-80 Tape Mod I and Mod III 16K 0136R **$14.95**

Everyday RUSSIAN

Now that you have the Russian alphabet down-pat with BEGINNER'S RUSSIAN (0136R), you can begin to speak EVERYDAY RUSSIAN. Quickly learn to read and speak necessary Russian words and phrases in the comfort of your home. You'll also learn the order of the Cyrillic alphabet for referencing in Russian dictionaries. Learning Russian couldn't be easier with this patient and never-tiring tutor.

TRS-80 Tape Mod I and Mod III 16K 0137R $14.95





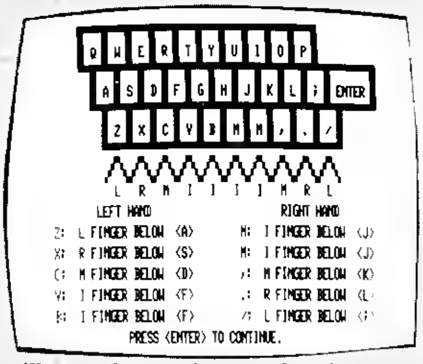
RUSSIAN Disk

Beginner's Russian and Everyday Russian are available together on one convenient disk.

TRS-80 Disk Mod I and Mod III 32K 0212RD **$29.95**

Apple Disk Applesoft 32K 0283AD **$29.95**

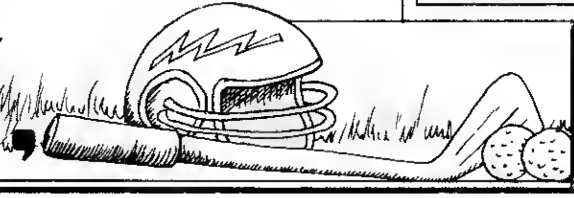
TYPING TEACHER



You'll never have to hunt-and-peck again! A seven-part tutorial allows you to advance at your own speed, providing plenty of space for practice without wasting paper. On-screen displays keep eyes up and away from the keyboard to develop proper technique. There's no teacher more patient than this program.

TRS-80 Tape Mod I and Mod III 16K 0099R **$17.95**

GAMES,



GOLF

Sit back and enjoy a rugged, challenging 18 hole game of golf! Get a complete choice of clubs and shooting angles—and let your computer keep score for you! For one or two players.

Apple Tape Applesoft 16K 0018A *$14.95*

CHIMERA

A legendary fire-breathing creature; or six fearsome arcade games for your Pet. Think quick and challenge your reflexes with DRAGON, DRAGON HUNT, DUNGEON, DROPOFF, REFLEX, and BATTER UP. Ages 7 through adolescence.

Pet Tape Old and New ROM 8K 0110P *$14.95*

PET DEMO I

7 exciting games—a new challenge for every mood. Gamble, shoot, race and chase as you enjoy SLOT MACHINE, CHASE, FLYING PHEASANT, SITTING DUCKS, CRAPS, GRAN PRIX 2001, and FOX AND HOUNDS. Ages 7 through adolescence. Arcade.

Pet Tape Old and New ROM 8K 0035P *$14.95*

TURF AND TARGET

Here's the perfect package for the sportsman! Catch all the action—even when it's 20 below—with QUARTERBACK, SOCCER II, SHOOT, and TARGET. Ages 8 through adolescence. Arcade.

Pet Tape Old and New ROM 8K 0097P *$14.95*

HOUSE OF 30 GABLES

Tread softly as you enter the mysterious House of Thirty Gables. Inside awaits golden treasure for the stout of heart. But beware! Serpents, dragons and trolls lurk within, ready to trap the unwary stranger. Adventure.

TRS-80 Tape Mod I and Mod III 16K 0219R *$14.95*

DUNGEON OF DEATH

Each step you take through the 12 levels of this diabolical maze brings you closer to the ultimate treasure. . . the Holy Grail. Hideous monsters and unspeakable dangers dog your every step. And once you find your treasure, you'll find yourself face-to-face with the most fearsome Smaug. Only those with supreme courage dare enter the DUNGEON OF DEATH. Adventure.

PET Tape Old and New ROM 8K 0064P *$14.95*

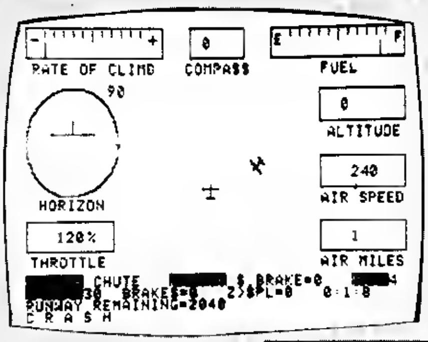
NIGHT FLIGHT

Dangerous photo-recon missions await as you wing your way through the murky night with only your instrument panel to guide you. This program lets you takeoff, land, and fly in a simulation so real we've included the basic principles of flight in the instructions.

TRS-80 Tape Mod I and Mod III 16K 0117R *$14.95*

Apple Disk Applesoft 32K 0304AD $24.95

JET FIGHTER PILOT



Takeoff, intercept the enemy, navigate to the airport or carrier, and land. Jet Fighter Pilot takes you as close to real combat flying as possible. . . without pulling G's. A challenge you will never outgrow—Jet Fighter Pilot offers you more commands and controls than any other flight simulation we know of! Simulation.
TRS-80 Tape Mod I and Mod III 16K 0159R *$14.95*
Apple Disk Applesoft 32K 0329AD *$24.95*

Z-80 CHECKERS

An excellent version of computer checkers that has some outstanding features unavailable elsewhere: seven different skill levels, a board Editor, a practice mode, and more!
TRS-80 Tape Mod I & Mod III 16K 0310R *$14.95*

MUSIC MASTER

Turn your computer into an electronic organ and play music using this quartet of programs.
MICRO ORGAN Your computer keyboard becomes an organ keyboard.
KALEIDOPY A combination of a kaleidoscope and a player piano. . . Watch while the computer creates a pattern and plays it as music.
COMPOSER Generate quasi-random music with a repeating refrain.
KEYMANIA Compete to reproduce the greatest number of notes generated by your computer.
TRS-80 Tape Mod I and Mod III 4K 0084R *$14.95*

MIMIC

A fast action memory game that tests your concentration and reflexes. Your computer flashes number sequences and you repeat them within the time limit. With three skill levels and sound effects option.
TRS-80 Tape Mod I Only 16K 0066R *$14.95*
PET Tape Old and New ROM 8K 0039P *$14.95*
Apple Tape 16K 0025A *$14.95*

HOOPTEDOODLE

8 entertaining programs for you and your PET to enjoy. Among the many feats you'll be challenged to accomplish are to. . . make your way through a seemingly endless maze and avoid the monster within, cross treacherous mine fields, and fly dangerous bombing missions. Great for children. Arcade.
PET Tape Old and New ROM 8K 0091P *$14.95*

Gomoku & 3D Tic Tac Toe

A dynamic duo of popular games: GOMOKU, the ancient oriental board game of strategy and a three-dimensional version of tic-tac-toe. Choose among different board sizes for even more challenge!
TRS-80 Tape Mod I & Mod III 16K 0210R *$14.95*
PET Tape Old and New ROM 8K 0038P *$14.95*

ARCADE I

Here's a package sure to please whether you like your fun in or out-of-doors.
KITE FIGHT You'll find high-flying excitement in this computer version of a sport the people of Japan and India have enjoyed for centuries.
PINBALL combines the superb graphics of the PET and the classic features of arcade pinball to provide a game so real, all you'll miss is the TILT. Audio amplifier and speaker required for sound.
PET Tape Old and New ROM 8K 0074P *$14.95*

ARCADE II

Challenge your memory and sharpen your reflexes with this package that offers something for everyone. Catch the elusive UFO before it gets away. HIT the target, if you can aim while your moving. BLOCKADE Skill and strategy are necessary to block your opponent's path without letting him block yours.
PET Tape Old and New ROM 8K 0045P *$14.95*

PERFECT PONG

Ping Pong has entered the space age! Turn your computer into a fast-paced action machine. Hit the "ball" with your paddle—don't let it disappear off the screen! Choose from 8 different game boards. Arcade
TRS-80 Tape Model I only 16K 0120R *$14.95*

TEMPLE OF THE SUN

You have discovered the ancient ruins of the long lost Temple of the Sun. Hidden in the underground chambers are treasures beyond your wildest dreams and the key that unlocks the magical powers of the Shamon, an order of supremely powerful wizards. Adventure.
TRS-80 Tape Mod I Only 5012R *$19.95*
TRS-80 Disk Mod I Only 5011RD *$29.95*
TRS-80 Disk Mod III 5013RD3 *$29.95*

Domes of Kilgari

You are doomed to the lonely wasteland of the planet Kilgari forever—unless you can penetrate the Digitron Ion Station, (which is programmed to destroy all intruders) and retrieve an Ion rod to replenish your fuel supply. Puzzles and mazes abound in this high-caliber adventure.
TRS-80 Tape Mod I Only 16K 5014R *$19.95*
TRS-80 Disk Mod I Only 32K 5015RD *$29.95*
TRS-80 Disk Mod III 32K 5016RD3 *$29.95*

THE BUSINESS GAME

Play the game of tough-minded business persons. This game simulates an actual period in U.S. economic history; you manage your business by making hard decisions, bargaining, competing and (hopefully) succeeding! It's fast, challenging, rewarding and entertaining. Let us give you the Business!
TRS-80 Tape Mod I & Mod III 16K 0158R *$14.95*

SWAMP WAR

You are stranded on a wet and definitely unwonderful waterworld inhabited by some pretty slimy swamp creatures. Your only hope is to get the parts from the abandoned transporters located on the nearby islands to make repairs. This might not be an easy task, seeing as how a lot of crud and creatures stand between you and these spare parts. Do your best to rid the islands of their inhospitable inhabitants, collect all transporters, and get outta town!
Joysticks optional.
TRS-80 Tape Mod I and Mod III 16K 0312R *$14.95*

ALL STARS

A collection of our most popular games.
SANTA PARAVIA (see program #0043R) Take the reins of this medieval city-state and control the lives of the serfs who produce the wealth of the realm. Your title and domain will increase if you rule well.
OIL TYCOON (see program #0023R) Two players compete to become top oil magnate in this game of strategy and luck.
PARADISE TRADER As captain of a trading schooner, you sail your ship about the Carribean acquiring valuable cargo, while you attempt to avoid pirates, ghost ships, and hurricanes.
MILLIONAIRE Can you turn $1000 into $1,000,000 in 15 years? It depends on your financial finesse in business dealings as you buy and sell properties, negotiate bank loans, collect rentals and accept bids.
TIMBER BARON Shrewd strategy is the key to your success as you attempt to avoid the many hazards of this profitable industry.
BATTLEGROUND (see program 0141R) You and your opponent command Allied and Axis forces in this World War II simulation. Find out if you're the stuff generals are made of.
TRS-80 Disk Mod I and III 32K 0213BD *$34.95*

WHO DUN IT?

Two highly entertaining games that challenge your deductive powers. In WHO DUN IT? you're the crack detective who must investigate and solve any of five dastardly crimes. The second game, EDUCATION, is a game of logic that tests your deductive skills!
TRS-80 Tape Mod I & Mod III 16K 0047R
$14.95

MOUNTAIN PILOT

The fast buck can still be made... but there's a price. If you can bring supplies to the desperate miners of Goldtown, you'll be paid well. Your return trip with the gold boullion is the real payload though. The catch? Can you make it through treacherous Eagle Pass not once, but TWICE?! Simulation
TRS-80 Tape Color Computer/Extended BASIC 0370RC 16K *$19.95*

MENTAL GYMNASTICS

Challenge your mind with these ancient games.
REVERSI—An Othello[11]—type game
WARI—A real test of mental concentration.
Adolescence through adult. Strategy.
Heath Tape Benton Harbor BASIC 8K
0087H *$14.95*

OIL TYCOON

Test your skill at money making as an oil magnate. Strategy and fate decide whether you'll increase your millions by out-drilling and under-selling your competitor, or lose your fortune to an oil spill. Two players.
TRS-80 Tape Mod I & Mod III 16K
0023R *$14.95*
Apple Tape Applesoft 16K
0079A *$14.95*

Battleground

It's late 1944 and the Allied Forces are sweeping toward Berlin. As the commander of your sector, you control tanks, planes, artillery, infantry, engineers, bunkers, and vehicles. The battle map shows how you have chosen to deploy your forces. Shoot, attack, drop paratroopers, lay mines. For two players.
Strategy.
TRS-80 Tape Mod I and Mod III 16K 0141R
$14.95

Tower of Fear

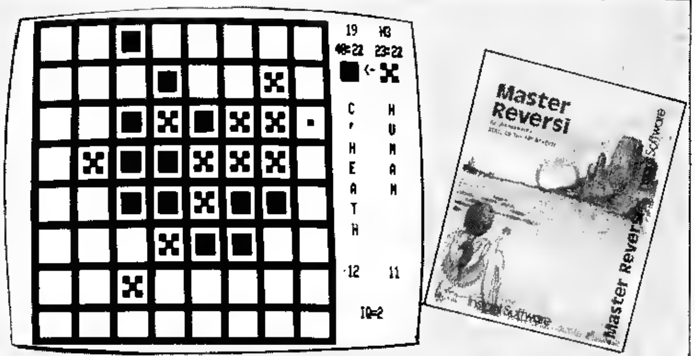
Enter the castle of Blackheart Flamethrower, the last of the magnificent wizards who's magical prowess has conquered even death. Many have tried to penetrate this fortress but none survive to tell the story of the treasures and terror that await therein. Adventure.
TRS-80 Tape Color Computer 16K 5020RC
$19.95

KID'S GALLERY

Here are 5 games that will keep the younger members of the family and their friends playing for hours.
3-D Tic-Tac-Toe You can play on the traditional 3 × 3 × 3 board or try the more difficult 4 × 4 × 4 × 4 version. Four levels of play.
Haunted House Escape the Haunted House by finding the right curse against resident ghosts, vampires and werewolves.
Shoot-out You and a friend compete to see who's fastest on the draw.
Frog Mountain Help Freddy the frog catch his lunch of bugs and butterflies and rack up your points.
Vegas Here's a solt machine you and your friends can play without putting in a penny.
TRS-80 Tape Mod I Only 16K 0172R *$14.95*

BALL TURRET GUNNER

Fight back against the Petro Giants that threaten your existence! Enter the elite service of the BTG; maneuver your laser cannon; destroy the invading gnat ships—just hope that you're equipped to deal with conditions in your space-zone. Fast action that's part of a tense interstellar drama! Arcade. Sound.
TRS-80 Tape Mod I and Mod III 16K 0051R
$14.95



Master Reversi

Overcome all barriers standing between you and an Othello championship. Match your wits against a topnotch, tournament-winning program. "Reversi" is an ancient board game that challenges you to bracket your opponent's piece and reverse them so your color shows. Master Reversi is special because it allows you to analyze your moves as well as the computer's moves and the games in amazing depth. No other program offers so many ways to improve your game. Master Reversi will challenge you, teach you and bring you innumerable hours of fun, no matter what your degree of expertise!
TRS-80 Tape Model I and III 16K 0417R *$19.95*

MASTER REVERSI—Disk Version

All the features described for Master Reversi are contained on the disk PLUS you get a library of tournament games that you can study and even replay.
TRS-80 Disk Models I only 16K 0378RD *$24.95*

Games

IQ TEST

There aren't too many of us who can justifiably claim to be a genius, but here's your chance to find out if your hunch is right. IQ Test will administer and score an intelligence test in just 30 minutes. While we wouldn't suggest you use this program in place of the tests administered by trained psychologists, you'll find them sufficiently accurate to be fun and entertaining.

TRS-80 Tape Mod I and Mod III 16K 0157R *$14.95*

MINDWARP

Two devilishly-intricate games for the math-minded: MIND TWIST gives you a number series for which you must determine the correct formula; and MIND BENDER is a multi-level code game in which you must guess the next number in your computer's "secret" numeric sequence. The game theoretician's delight!

TRS-80 Tape Mod I & Mod III 16K 0118R *$14.95*

DRAGONQUEST

The princess has been captured and you, faithful knight, must safely deliver her from the clutches of the most fearful Smaegor, monarch of dragons. Locate the weapons, charms, and tools you need to accomplish your task, but act quickly, for Smaegor dines when the sun sets. Adventure.

TRS-80 Tape Mod I and Mod III 16K 5006R *$15.95*

TRS-80 Disk Mod I and Mod III 32K 5010RD *$21.95*

INVESTOR'S PARADISE

Experience all the thrills and triumphs of the stock market without risking a dime.

STOCK TREK Can you transform $5000 into a fortune in just 12 short months? Only if you possess the financial savvy to invest wisely and read the market right. Play with up to 5 other investors.

SPECULATION Short of actually investing real cash, this is the closest you can come to playing the stock market. Start with up to $10,000 and an equal amount deposited in a savings account, then make your money work for you. Big risks can mean big gains, and just as well, big losses.

Both simulations are a great way to learn all about the stock market and preview your money-making abilities before you attempt the real thing.

TRS-80 Tape Mod I and Mod III 16K 0125R *$14.95*

DANGER IN ORBIT

Nerves of steel and lightning reflexes will enable you to blast the alien ships and destroy the asteroids! Watch out! Your antimatter cannon fire will split the asteroids into smaller and smaller bodies that can still destroy your spaceship. The action game for action-lovers! Arcade. Sound. Joysticks optional.

TRS-80 Tape Mod I and Mod III 16K 0237R *$19.95*

TRS-80 Disk Mod I and Mod III 16K 0247RD *$24.95*

Air Flight Simulation

Takeoff, land, use flight instruments and perform daring aerial stunts. This simulation reproduces all the basic flying characteristics of a light aircraft. Do your crashing now—before you try the real thing!

TRS-80 Tape Mod I and Mod III 16K 0017R *$14.95*

Apple Tape Applesoft 16K 0148A *$14.95*

RAMROM PATROL

The Space Age has begun! But with new universes to explore come new dangers to combat in this package of exciting children's games.

RAMROM PATROL Defend the earth by shooting down the enemy ships.

TIE FIGHTER Man your X-wing fighter to destory the Empire's dreaded Tie Fighters.

KLINCON CAPTURE Capture the newest battle cruiser, the pride of the Klingon's fleet, without destroying it!

TRS-80 Tape Mod I and Mod III 16K 0028R *$14.95*

Dr. Chips

Talk to a brilliant psychoanalyst—he'll listen empathetically and respond on your computer screen. He can coax your innermost feelings out of you. Dr. Chips is also a great icebreaker at cocktail parties and a natural way to introduce your noncomputer friends to the computer.

TRS-80 Tape Mod I and Mod III 16K 0218R *$14.95*

Apple Disk Applesoft and Integer 32K 0254AD *$19.95*

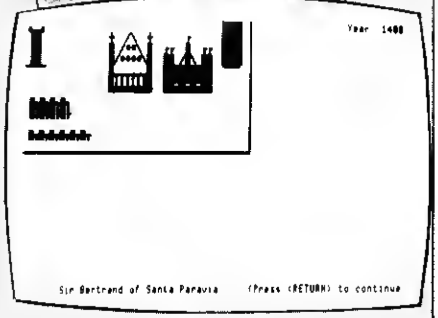
DOODLES & DISPLAYS II

Six creative graphics programs in one package: DOODLE PAD lets you draw pictures and save them on tape; SYMMETRICS is a computer kaleidoscope; DRAWING is for the serious computer artist; RANDON PATTERN DISPLAY makes your computer do the artwork; MATH CURVES brings geometry lessons to life; and RUG PATTERS helps you design rugs and just about anything else!

TRS-80 Tape Mod I only 16K 0042R *$14.95*

See page 14 for DANGER IN ORBIT, page 15 for ALIEN ATTACK FORCE and SANTA PARAVIA AND FIUMACCIO, page 16 for SKYBOMBERS, KITCHEN SINK and FLIGHT PATH.

Santa Paravia and Fiumaccio



You control grain harvests, set tax rates, and exercise justice in your attempt to increase your title and turn this insignificant city-state into a powerful kingdom. A classic strategy game that teaches modern economics in a medieval setting.
TRS-80 Tape Mod I & Mod III 16K 0043R *$14.95*
Apple Tape Applesoft 48K.0174A *$14.95*
Pet Tape Old and New ROM 16K 0175P *$14.95*
TI 99/4 Tape 16K 0273TI *$14.95*
Atari Tape 800 32K 0353AT *$14.95*

SPARROW COMMANDER

You have read about those brave men defending their ancient castle with a Kitchen Sink (see program #0386RD) and other oddments. Now the other side of the story can be told. Choose your strategy to guide your troops safely to their nesting ground amidst the barrage of shells, safes, and even kitchen sinks that their merciless enemies, the Cloud People, rain down upon them. As fearless leader of the sparrows, you can make the difference between sparrow survival and bird obliteration. Now you can experience the feeling of raw power as the brain behind the birds—the brave SPARROW COMMANDER. Arcade. Sound.
TRS-80 Disk Mod I and Mod III 32K 0387RD *$19.95*

MINOTAUR

Go far into the future to meet the past! The Labyrinth of ancient Crete has been recreated by the diabolic king of the planet of New Crete. The intended victims are your clones!—Your only source of wealth and prestige! Become a Lord Citizen of the forbidding planet. Strategy. Sound.
TRS-80 Tape Mod I and Mod III 16K 0318R *$14.95*

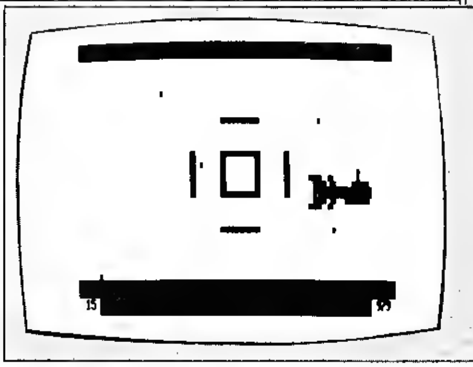
LIFE

Creation made easy! With this educational program your computer becomes a mini-laboratory. You can study the life cycle of cellular colonies choosing the formations you wish to observe, and even experiment with cell structures of your own invention. Ages 12 to adult.
TRS-80 Tape Mod I Only 16K 0078R *$14.95*

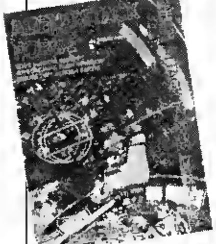
COSMIC PATROL

A quick wit and reflex are required by players in this space odyssey where you must do unto others before they do unto you. For each Quelon space ship you miss, precious units of your energy supply will be drained resulting in the loss of life and limb of said player. Fast real-time action, optional sound and remarkable graphics put this machine language program at the head of its class. Arcade. Joysticks optional.
TRS-80 Tape Mod I and Mod III Sound Option 16K 0223R *$19.95*
TRS-80 Disk Mod I and Mod III Sound Option 16K 0224RD *$24.95*

PADDLE FUN



Four action-filled games challenge your prowess.
ALIEN ATTACK FORCE Destroy the 55 invading flying saucers before they destroy your antimatter gun.
SPACE WARS Try to disable your friends flying saucers. . .
HOWITZER Aim accurately and choose the optimum firing velocity. Destory the enemy's gun!
Apple Disk Applesoft 32K 0163AD ***$19.95***



ALIEN ATTACK FORCE

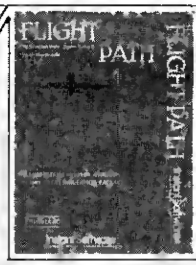
With all of Earth's defenses dead, you are the last vestige of hope for the planet as hundreds of armed invaders make their way to attack. Nine levels of difficulty continue to challenge you as your skills increase. Arcade. Joysticks optional.
TRS-80 Tape Mod I and Mod III 16K 0240R *$14.95*

See page 12 for SWAMP WARS and JET FIGHTER PILOT, Page 13 for BALL TURRET GUNNER and MASTER REVERSI, page 14 for AIR FLIGHT SIMULATION, page 15 for COSMIC PATROL and, page 16 for SPACE SHUTTLE.

SPACE SHUTTLE

Experience the Ultimate Flight. Command the next launch of the space shuttle Columbia: its orbit, reentry and landing procedures are all in your control. A NASA engineer says this simulation is quite close in design to the one astronauts use for training!

TRS-80 Tape Mod I & Mod III 16K 0332R *$19.95*



FLIGHT PATH

These three tantalizing flight games will whet your appetite for aviation:

MOUNTAIN PILOT Fly over Eagle pass. (See package #0362AD).

PRECISION APPROACH RADAR Land a UFO! (See package #0362AD).

O'HARE Be in charge of the control tower of a busy airport. Direct the speed and approach of 20 planes. The safety of each person aboard depends on your skill! Simulation.

TRS-80 Tape Mod I and Mod III 16K 0171R *$14.95*

KITCHEN SINK

As commander of the castle-in-the-clouds, you must destroy the invading sparrows using everything you have, including the kitchen sink. Plan your strategy wisely: you may shoot, bomb, and ram with maneuverable boats. But your weapons and your time are limited! The perfect fast-action strategy game for those who aren't satisfied with mindless shooting. Arcade. Sound.

TRS-80 Disk Mod I and Mod III 32K 0386RD *$19.95*

AIRMAIL PILOT

Fly your Jenny biplane from Columbus to Chicago before the deadline. You need only avoid running out of gas, being struck by lighting, crashing in a down draft, missing the runway..... Challenge your reflexes. Remember, the mail must go through!

TRS-80 Tape Mod I and Mod III 16K 0106R *$14.95*

T.I. Tape T.I. 99/4 16K 0274TI $14.95

DAREDEVIL

Six super-fast arcade games that challenge your skill and stamina. Be a ski racer, a bob sledder, an anti-aircraft gunner and get a choice of three brilliant auto racing programs. Up to 10 people can play! With sound effects option.

TRS-80 Tape Mod I & Mod III 16K 0082R *$14.95*

In reference to MISSION-MUD (0235RD) . . .

"Grand-kids love it."

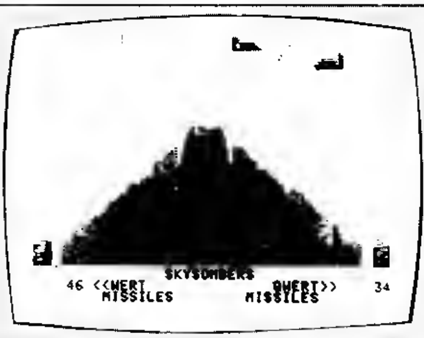
Gerry Gervais
Morra NY 12957

SKYBOMBERS II

Air warfare becomes vivid reality as you and an opponent command fighter bombers against each other. You must first fly over the treacherous mountain that separates your countries before you bomb the enemy blockhouse into oblivion—that is if you're pilot enough to escape enemy fire along the way. Game paddles required. Arcade. Sound.

Apple II Tape Applesoft and Integer 32K 0183A *$14.95*

Apple II Disk Applesoft and Integer 32K 0271AD *$24.95*



Mission: MUD

Defend the Earth base city by destroying the approaching Mud Monsters and the slimy Mud Patch they inhabit. Mission: Mud is not just another arcade game. Like a chess game in which the pieces have weapons, it combines the challenge of strategy with the excitement of chance. Arcade.

TRS-80 Disk Mod I and Mod III 32K 0325RD $19.95

apple fun

Five fun-filled programs provide amusement and challenge for every family member.

MIMIC How good is your memory? Your computer will display a series of graphic figures you must be able to repeat. Five levels of difficulty for continuous challenge.

AIR FLIGHT SIMULATION You're flying blind, depending solely on your instrument panel. Watch closely, it's your only clue to your position as you take off, climb, descend, and land in this realistic simulation.

COLORMASTER This brainteaser will sharpen your power of deduction as you attempt to guess the four colors and the sequence chosen by your opponent or the computer.

STAR SHIP ATTACK Defend the orbiting supply satellites from the invading enemy or the planet is doomed!

TRILOGY A fascinating contest of logic, based on the game of tic-tac-toe, using colors and a three-dimensional grid. Compete with a friend or the computer. Arcade.

Apple II Disk Applesoft 32K 0161AD *$19.95*

FLYING CIRCUS

Seven favorite flight programs combine to excite aviation enthusiasts.

AIR FLIGHT SIMULATION Fly a light plane. (See package #0017R.)

AIRMAIL PILOT Get the mail through. (See package #0106R.)

MOUNTAIN PILOT Fly over Eagle pass. (See package #0362AD.)

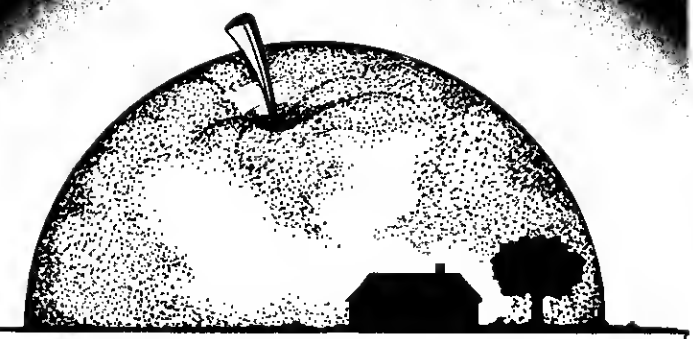
JET FIGHTER PILOT Intercept the enemy. (See package #0159R.)

PRECISION APPROACH RADAR Land a UFO! (See package #0362AD.)

O'HARE Guide the pilots in safely. (See package #0171R.)

TRS-80 Disk Mod I only 32K 0205RD ***$39.95***

Finally, a magazine as good as your Apple*



From Wayne Green, Inc.—publishers of *80 Micro, Microcomputing* and *Desktop Computing*—

comes. . . **inCider**

inCider premiers this winter and *will expand the limits of your Apple like 80 Micro has blown the lid off the TRS-80.*

Not just another Apple magazine-but a comprehensive system specific monthly filled with. . .

- programs
- software applications
- hardware modifications
- reviews
- new product announcements
- advertising
- tutorials
- games

. . . written by technically competent editors and columnists. *inCider* is designed to provide you with the *newest* and *best* for your Apple.

Want to know the easiest way to get the kinks out of your programs? *inCider* will show you. Want to know which peripherals have the best history and the brightest future? *inCider* will tell you. Want to expand your knowledge of hardware? Or become an expert programmer? Want to discover which word processors give you the most for your money? Or how your Apple can better manage your financial affairs? *inCider* will answer these and many other questions each month.

The inaugural issue of *inCider* will be available in early December. And you can become a Charter Subscriber to *inCider* for only $19.97! Fill out the attached reply card and drop it in the mail, or use the coupon below. Or call toll free

1-800-258-5473

And here's a special bonus offer: enclose payment (check or credit card #) with your order and you'll receive a 13th issue of *inCider* absolutely free!

Be an *inCider* and get more useable information for your Apple than you ever dreamed possible.
Subscribe to *inCider* today.

*Apple is a trademark of Apple Computer Inc.

---

inCider **YES,** I want to be an inCider. Send me 12 issues for only $19.97.

☐ CHECK ☐ MC ☐ VISA ☐ AE ☐ BILL ME ☐ I have enclosed payment-please put me down for my free 13th issue.

Card# ______________________

Interbank ______________ Exp. date ______________

Signature ______________________

Name ______________________

Address ______________________

City ______________ State ______ Zip ______

Canada and mexico—$22.97, 1 year only U.S. funds
Foreign Surface—$39.97, 1 year only U.S. funds drawn on U.S. bank
Allow 6-8 weeks for delivery

inCider
PO Box 911
Farmingdale, NY 11737

331CSW

DEALERS UNDER THE SUN ACROSS THE U.S.A.

ALABAMA
Anniston, KEMP'S DEFICENTER
Birmingham, THE COMPUTER CENTER
Florence, ANDERSON COMPUTERS
Homewood, VILLAGE COMPUTERS
Huntsville, ANDERSON COMPUTERS
Mobile, OLENSKY BROTHERS
Stapleton, RADIO SHACK
Tuscaloosa, THE COMPUTER ROOM, INC
ALASKA
Anchorage, COMPUTER TALK
Juneau, JUNEAU ELECTRONICS
ARIZONA
Mesa, MESA ELECTRONICS
Mesa, MILLET'S ELECTRONICS
Mesa, PERSONAL COMPUTER PLACE
Phoenix, C E R F
Phoenix, COMPUTER STORE
Phoenix, KINGS MICRO ELECTRONIC SPEC
Phoenix, PROFESSIONAL DATA SYSTEMS
Safford, M & M ELECTRONICS
Sierra Vista, ADVENTURE HOBBIES
ARKANSAS
Conway, S.I.S., INC
Hot Springs, MICRO COMPUTER SYSTEMS, INC
Springdale, CAPPS CLINIC
Sun City, RUSALEM ELECTRONICS
Tucson, SIMUTEK COMPUTER PRODUCTS
CALIFORNIA
Albany, ALBANY TYPEWRITER SERVICE
Bellflower, THE SOFTWARE AFFAIR
Camarillo, COMPUTER HORIZONS
Chatsworth, OPPORTUNITIES FOR LEARNING
Chino, SILVER SPUR ELECTRONICS
Citrus Heights, BYTE SHOP
Citrus Heights, SOFTWARE PLUS
Commerce, THE FEDERATED GROUP
Corcoran, HUNTINGTON COMPUTING
Covina, G & H ELECTRONIC SUPPLY
Cupertino, WESTERN MICROTECHNOLOGY, INC
Cypress, SCR ELECTRONICS
Davis, THE STATION
Diamond Bar, COMPUTER MART OF CALIFORNIA
El Cajon, RADIO SHACK
El Cerrito, COMPUTERLAND
El Cerrito, PC COMPUTERS
El Monte, CHASO COMPUTERS
El Toro, WABASH APPLE
Encinitas, FAMILY COMPUTING, INC
Fortuna, R & V SOUND (R/S)
Garden Grove, WARRIORS, WIZARDS & ROBOTS
Grass Valley, ADVANCED RADIO (R/S)
Half Moon Bay, STRAWFLOWER ELECTRONICS
Hawthorne, JADE COMPUTER PRODUCTS
Hayward, BYTE INDUSTRIES
Hayward, ON LINE MICRO CENTERS
Huntington Beach, THE SOFTWARE STORE
Irvine, COMPUTER SOFTWARE BOOKS R US
Lakewood, THE SOFTWARE STOP
Lancaster, COMPUSUP
Lawndale, COMPUTERLAND SOUTH BAY
Lompoc, GOOD FAIRE NATURAL FOODS
Long Beach, ANIS COMPUTERS
Long Beach, BALBOA GAME COMPANY
Los Altos, SOFTWARE EMPORIUM
Los Angeles, OPAMP/TECHNICAL BOOKS
Los Angeles, THE SOFTWARE STORE
Malibu, MALIBU MICROCOMPUTING
Mammoth Lakes, COMPUTER HOUSE
Manteca, CLARKE BOOKS & COMPUTERS
Morro Bay, COAST ELECTRONICS
Mountain View, BYTE SHOP
Northridge, H & W ELECTRIC
Orange, COMPUSOUND
Palm Springs, RADIO SHACK
Palmdale, COMPUTERS MADE EASY
Pleasant Hill, ASHER & ASSOCIATES
Poway, COMPUTER ALLIANCE CO
Poway, RADIO SHACK
Riverside, INLAND ELECTRO-MART
Sacramento, CAPITAL COMPUTER SYSTEMS
Sacramento, KBL MICRO SYSTEMS
Salinas, SALINAS HOBBY CENTER
San Diego, COMPUTER ALLIANCE COMPANY
San Diego, COMPUTER MERCHANT
San Diego, COMPUTERLAND
San Diego, DIMENSIONAL SOFTWARE
San Diego, SOUND HOUSE
San Francisco, COMPUTERLAND
San Francisco, STACEY'S BOOKSTORE
San Francisco, THE SOFTWARE CENTER
San Jose, ALLTRONICS
San Jose, COMPUTERS & COMPUTING SOL
San Jose, R.A.C. PRODUCTS
San Jose, SHAVER RADIO, INC
San Leandro, COMPUTER STORE
San Luis Rey, COMPUTER INFORMATION EXCH.
Santa Ana, ADVANCE COMPUTER PRODUCTS
Santa Barbara, GAME-O-RAMA
Santa Clara, AFFORDABLE COMPUTER SYSTEMS
Santa Fe Spring, COMPUTER FORUM
Santa Monica, THE COMPUTER STORE
Signal Hill, ASAP COMPUTERS
Simi Valley, S P D S COMPUTER SYSTEMS
Stanton, STARLINE DISCOUNT HOBBIES
Sunnyvale, COMPUTER PLUS, INC.
Tarzana, MICRO BUSINESS WORLD
Torrance, NET PROFIT COMPUTERS
Upland, FOOTHILL COMPUTER CENTER
Walnut Creek, INFOMAX
Weaverville, DeROSEAR APPLIANCE & REPAIR
Westminster, COMPUTER COMPONENT
Westminster, COMPUTER WAVE, INC
Westminster, COMPUTER WORLD
COLORADO
Arvada, COMPUTERLAND-NORTH DENVER
Boulder, COMPUTERLAND
Denver, APPARAT, INC.
Denver, EISTEL'S MICRO ELECTRONICS
Englewood, VIDEO WORLD
Fort Collins, COMPUTER CONCEPTS OF COLORADO
Fort Collins, POOR RICHARD'S CALCULATORS
Pueblo, COMPUTER SHACK
Westminster, COLORADO COMPUTER SYSTEMS
CONNECTICUT
Bethel, TECHNOLOGY SYSTEMS
Fairfield, COMPUTERLAND
Greenwich, COMPUTER CORNER II
Groton, AMERICAN BUSINESS COMPUTER
Hamden, COMPUTERLAND
Manchester, INSTRUCTIONAL SYSTEMS COMPUTE
New Britain, ACTION COMPUTERS
New London, COMPUTERLAB
Newtown, THE SOFTWARE SOURCE
Southington, AM COMPUTER PRODUCTS
Stamford, COMPUTER STORE
Westport, COMPUTER WORKS, INC
DELAWARE
Newark, COMPUTERLAND
Wilmington, MICRO PRODUCTS
DISTRICT OF COLUMBIA
Washington, THE PROGRAM STORE
FLORIDA
Bradenton, NEW WORLD COMPUTERS
Clearwater, COMPUTER WORLDS
Clearwater, RAY'S AMATEUR RADIO
Fort Myers, COMPUTERLAND OF FORT MYERS
Fort Pierce, YOUR BASIC COMPUTER STORE
Fort Lauderdale, COMPUTER JUNCTION
Fort Lauderdale, COMPUTERLAND
Fort Lauderdale, FOCUS SCIENTIFIC
Fort Lauderdale, SOFTWARE CONNECTION
Gainesville, COMPUTER SYSTEM RESOURCES
Gainesville, MIKE'S BOOKSTORE
Gainesville, SOUND IDEAS
Gainesville, XEROGRAPHIC BUSINESS STORE
Hollywood, COMPWARE
Jacksonville, COMPUTERLAND OF JACKSONVILLE
Jacksonville, POCKET POWER
Jacksonville, SAV-A-BYTE
Jacksonville, WILLIAMS RADIO & TV
Longwood, ADVENTURE INTERNATIONAL
Melbourne, H I S COMPUTERMATION
Miami, COMPUTER VILLAGE
Miami, EAST WINGS CORP.
Miami, VINANCE
Ocala, DOSSEY'S HOME COMPUTER SOFTWARE
Ocala, RICK'S TV
Orlando, CREATIVE COMPUTER CENTER
Orlando, SOUTH EAST MICRO DATA
Ormond Beach, VANN DATA SERVICES
Panama City, COMPUTER SYSTEMS GROUP
Pensacola, KOBY'S CORNER
Plantation, TURNKEY MINI COMPUTER
Sarasota, COMPUTERLAND
Sarasota, H & H HOBBY SALES
Tampa, AMF MICROCOMPUTER CENTER
Tampa, COMPUTERLAND
Tampa, MICROCOMPUTER SYSTEMS, INC
Tampa, THE COMPUTER SHOP
Tampa, THE MICROCOMPUTER CENTER
West Melbourne, COMPUTER HAVEN
West Palm Beach, COMPUTER CENTER
Wintergarden, ALL SYSTEMS COMPANY
GEORGIA
Atlanta, ATLANTA COMPUTER MART
Augusta, BAILEY'S COMPUTER SHOP
Perry, LAWHORN ELECTRONICS
Savannah, CHRIS & DAD'S HOBBIES
Savannah, LOGICAL CHOICE
Smyrna, COMPUTERLAND
Warner Robins, COMPUTER SERVICES CO
Wrens, FLEMING DRUG CO
HAWAII
Honolulu, HONOLULU ELECTRONICS
Honolulu, MEGABYTE SYSTEMS
Honolulu, RADIO SHACK
Lahaina, MILLS ELECTRONICS, INC.
IDAHO
Boise, ELECTRONIC SPECIALTIES
Buhl, IDAHO MICROCOMPUTER
Fruitland, DENNIS STONE ENTERPRISES
Idaho Falls, R & L DATA SYSTEMS
Moscow, COX & NELSON
ILLINOIS
Aurora, FARNSWORTH COMPUTER CENTER
Bensenville, JAMES M. CONWAY CORP.
Burr Ridge, J. W. HALM CONSTRUCTION CO
Charleston, CREATIVE PROGRAMMING
Chicago, DATA SYSTEMS
Chicago, GARCIA AND ASSOCIATES
Chicago, MULTI-MEDIA EDUCATIONAL RESOURCES
Chicago, TRIGON ELECTRONICS
Elmhurst, COMPUTER JUNCTION
Elmhurst, COMPUTERLAND OF ELMHURST
Evanston, CHICAGO MAIN NEWSTAND
Geneseo, RADIO SHACK
Glen Ellyn, COMPUTER CENTER
Glen Ellyn, MORSCHNETZEL COMPUTER STORE
Harrisburg, ILLINOIS CUSTOM COMPUTERS
LaGrange, BYTE SHOP
Lincolnwood, ROZEL INDUSTRIES
Lombard, MIDWEST MICRO COMPUTERS
Macomb, DON'S STEREO
Mundelein, COMPUTERLAND
Naperville, ILLINI MICRO COMPUTERS, INC
Niles, COMPUTERLAND
Northbrook, NORTHBROOK COMPUTERS
Oak Brook Terr, OAK BROOK COMPUTER CENTRE
Peoria, COMPUTERLAND
Peoria, WALLACE MICRO-MART
Rockford, ALPINE COMPUTER CENTER
Rockford, COMPUTER STORE
Salem, RADIO SHACK
Schaumburg, CON FAB COMPUTER PRODUCTS
Schaumburg, DATA DOMAIN
Washington, COMPUTERS FOR YOU
INDIANA
Angola, D & D ELECTRONICS
Bloomington, DATA DOMAIN
Bloomington, GAME HUT
Dyer, SMITH COMPUTER SYSTEMS
Elkhart, SIMONTON LAKE DRUGS
Evansville, A B C HOBBYCRAFT CO
Evansville, THE HAM SHACK
Fort Wayne, FIRST BYTE
Indianapolis, GAME PRESERVE
Indianapolis, THE BOARDROOM
Lafayette, DIGITAL TECHNOLOGY
Noblesville, HARBOURTOWN SALES
Pendleton, FALL CREEK ELECTRONICS
IOWA
Ames, BEACON ELECTRONICS
Ames, CYBERIA, INC
Bettendorf, COSMOS COMMUNICATIONS
Bettendorf, MEMORY BANK, INC
Carroll, SERNETT LEISURE CENTER
Des Moines, BUSINESS DATA PROCESSING
Dubuque, COMPUTERS PLUS
Marion, THE PARTSTORE
KANSAS
Eldorado, LA FORGES, INC.
Herington, CENTRAL KANSAS COMPUTERS
Wichita, AMATEUR RADIO EQUIPMENT CO
Wichita, GOSUB INTERNATIONAL
Wichita, HIGH TECHNOLOGY
KENTUCKY
Bremen, PERRY'S COMPUTER
Lexington, CBM, INC
Louisville, COMPUTER MAGIC, INC
Paducah, HEDDEN ENTERPRISES
LOUISIANA
Baton Rouge, ACME BOOK CO.
Lake Charles, COMPUTER CONCEPTS
Metairie, COMPUTER SHOPPE, INC
Shreveport, COMPUTER SERVICES OF SHREVEPORT
Shreveport, MICRO BUSINESS SYSTEMS
MAINE
Auburn, MAINE MICROSYSTEMS
Auburn, MID-MAINE COMPUTER CO.
Bangor, MAINE COMPUTRONICS
Caribou, MERLIN COMMUNICATIONS
Eastport, COMPUTERWARE EAST
Fryeburg, FRYEBURG COMPUTER CENTER
Gorham, DOWN EAST SOFTWARE
Sebago Lake, MAINE SOFTWARE
Portland, RADIO SHACK
Portland, RADIO SHACK
Waterville, HOME PORT COMPUTERS
MARYLAND
Baltimore, THE PROGRAM STORE
Columbia, ADCA
Frederick, SOFTWARE ETC.
Hagerstown, MICRO DATA SYSTEMS
Hancock, HANCOCK ELECTRONICS
Laurel, THE COMM CENTER
Lexington Park, BOWES BOOKS
Marlow Heights, WILLS COMPUTER STORE
Pikesville, JACK FIVES ELECTRONICS, INC
Silver Springs, COMPUTER AGE
MASSACHUSETTS
Acton, GAME SHOP, INC
Billerica, MISSION COMPUTERS
Burlington, COMPUTER STORE
Charlestown, COMPUTER CITY
Danvers, GAME TECH
Framingham, STAR COMPUTING
Gardner, KAYCO
Leominster, L & M COMPUTER PRODUCTS
Leominster, LEOMINSTER NEWS
Ludlow, NORTHEAST COMPUTER STORE, INC
Lynn, LAND OF ELECTRONICS
Middleboro, MIDDLEBORO MUSIC
Natick, SIMSIM, INC.
Northampton, HI FI EXCHANGE
Tewksbury, OMNITEK SYSTEMS
West Springfield, COMPUTER VILLAGE
Waltham, N E COMPUTER SOFTWARE CO.
Watertown, MICROCON
Westford, SMALL BUSINESS SYSTEMS
MICHIGAN
Ann Arbor, A M ELECTRONICS
Ann Arbor, LEARNING CENTER, LTD
Ann Arbor, NEWMAN COMPUTER EXCHANGE
Battle Creek, HOBBY HOUSE
Brooklyn, WEATHERWAX DRUGS
Clawson, COMPUTER MART
Detroit, COMIC KINGDOM
Dimondale, MID-MICHIGAN MEMORY
Farmington Hill, COMPUTER CONNECTION
Fenton, TRI-COUNTY ELECT & SOUND CO.
Flint, COMPUTER MART
Garden City, COMPUTER CENTER
Grand Blanc, COMPUTER CONTACT, INC
Grand Rapids, COMPUTERS AND MORE
Hazel Park, FERRIS RADIO
Kalamazoo, COMPUTER ROOM
Kentwood, COMPUTERLAND KENTWOOD
Lansing, ALTERNATE SOURCE
Lansing, MID-WEST COMPUTER CENTER
Livonia, LEVEL IV PRODUCTS, INC.
Midland, COMPUTRONICS
Muskegon, EIGHT BIT CORNER
Port Huron, MCS MARKETING
South Haven, GOLDEN ANVIL
Southfield, COMPUTERLAND
Troy, RAINBOW COMPUTERS
Utica, BYTE BY BYTE COMPUTER STORE
Warren, LYCEUM, INC
West Bloomfield, ALL FOR LEARNING
Ypsilanti, YE OLDE TEACHERS SHOPPE
MINNESOTA
Andover, PHOENIX COMPUTER
Brooklyn Center, ZIM COMPUTERS
Burnsville, DIGITAL DEN
Eden Prairie, THE CODE ROOM
Marshall, RURAL AMERICAN ENTERPRISES
Minneapolis, PERSONAL BUSINESS SYSTEMS
Minneapolis, THE COMPUTER PLACE
White Bear Lake, MINNESOTA SOFTWARE, INC
MISSISSIPPI
Jackson, C-COM
Jackson, SOFTWARE HOUSE
West Point, DYER'S, INC.
MISSOURI
Belton, SOFTWARE SHACK
Cameron, D'S
Clarence, HIRRLINGERS
Columbia, CENTURY NEXT COMPUTERS
Florissant, COMP-U-TURS
Joplin, CRC COMPUTERS
Joplin, HOUSE OF COMPUTERS
Kansas City, DIGITAL SYSTEMS CORP
Parkville, MOORE SOFTWARE
Sedalia, BARRETO ASSOC., INC.
Springfield, COMPUTER MART
St. Charles, UNITED COMPUTER STORES
St. Louis, COMPUTER STATION
St. Louis, MICRO AGE COMPUTER STORE

MONTANA
Billings, THE COMPUTER STORE
Crystal Lake, COMPUTER WORKS
Havre, MOUNTAIN DATA
Kallispell, THE COMPUTER PLACE
Missoula, CONSOLIDATED SERVICES
NEBRASKA
Alliance, HATCH COMPUTER CENTER
Battle Creek, APPLETREE SOFTWARE
Columbus, ADVANCED COMPUTER SYSTEMS
Lincoln, HOBBY TOWN EAST PARK
Omaha, COMPUTERLAND
Omaha, COMPUTERS WEST
Omaha, MICROAGE COMPUTER STORE
Omaha, MIDWEST COMPUTER CO., INC
Scotts Bluff, SCOTTSBLUFF OFFICE PRODUCTS
NEVADA
Las Vegas, CENTURY 23
Las Vegas, HOME COMPUTERS
Las Vegas, HURLEY ELECTRONICS
NEW HAMPSHIRE
Amherst, COVER CRAFT
Amherst, HILLSBOROUGH PUBLISHERS
Concord, BITS N BYTES COMPUTER CENTER
Conway, STURDIVANT & DUNN
Fremont, PAUL'S TV
Hudson, TUFTS RADIO & ELECTRONICS
Keene, COMP-U-CRAFT
Keene, RADIO SHACK
Manchester, INTERPRO
Merrimack, COMPUTER SHOPPE, INC.
Milford, E & G EMLOR
Nashua, ALL SOFTWARE, INC.
Nashua, COMPUTERLAND
Peterborough, INSTANT SOFTWARE, INC
Peterborough, RADIO SHACK
Portsmouth, PORTSMOUTH COMPUTER CENTER
Portsmouth, THINK TANK
Salem, COMPUTER TOWN
Stratham, MICROSOURCE
NEW JERSEY
Caldwell, SOFTWARE MART
Cherry Hill, COMPUTERLAND
Cherry Hill, OMNIFAX
Cherry Hill, RADIO SHACK
Chester, MORRIS COUNTY STATIONERS
Denville, LASHEN ELECTRONICS
Dumont, ACADEMIC SOFTWARE
Dumont, H. L. STRICKLING
Englishtown, COMPUTER MADNESS
Fort Lee, SILENT PARTNER
Freehold, RADIO SHACK
Glassboro, ABE'S TV SALES & SERVICE
Hackettstown, T & J ELECTRONICS
Iselin, COMPUTER MART OF NJ, INC
Lodi, CONTEK
Mantua, ELECTRONIC WORLD
Maple Shade, GSB ELECTRONICS, INC.
Marlton, MIDAS DATA SYSTEMS, INC.
Marmora, OUTPOST (R/S)
Medford, CHANNEL ONE
Moorestown, RADIO SHACK
Paramus, COMPUTERLAND
Parsippany, ARTISAN ELECTRONICS
Pennsville, DAVE'S ELECTRONICS
Pine Brook, SOFTWARE CITY
Pitman, COMPUTERMANIA
Pompton Plains, COMPUTER CORNER OF NJ
Princeton, COMPUTER ENCOUNTER
Redbank, COMPUTER FORUM
Redbank, FELICE'S FOLLIES
Redbank, HOBBYMASTERS
Shrewsbury, MONMOUTH COMPUTER SERVICE
Somerset, RADIOS UNLIMITED
Sparta, MOUNTAIN ELECTRONICS
Teaneck, SOFTWARE CITY
Union City, ADELMAN'S STATIONERY CO.
Upper Montclair, SOFTWARE NETWORK
Voorhees, HOBBIES LIMITED STORE #203
West Milford, COMPUTER DISCOUNT OF AMERICA
West Trenton, BARGAIN BROTHERS
Westfield, COMPUTER DIMENSION
Whitehouse Station, CROWLEY'S
Woodbridge, THE GAME ROOM
NEW MEXICO
Alamogordo, THOMAS E. CARR JEWELER
Albuquerque, AUTEL ELECTRONICS CO
Albuquerque, MICROAGE COMPUTER STORE
Albuquerque, WARGAMES WEST
Carlsbad, MITCHELL MUSIC
Clovis, J & W ENTERPRISES
Las Cruces, SOUTHWEST COMPUTER CENTER
NEW YORK
Albion, ALMCRAFT RADIO SHACK
Brooklyn, MICRO HUT
Carle Place, COMPUTERLAND OF NASSAU
Cohoes, 80-MICROCOMPUTER SERVICES
Cooperstown, RADIO SHACK
De Witt, OMNIFAX
Dix Hills, APPLIED SOFTWARE
Endwell, COMPUTER TREE, INC.
Flushing, STRAIGHTLINE MARKETING
Fredonia, BITS & BYTES
Jericho, PROGRAMS UNLIMITED
Kingston, COMPUTER SHOP
Lynbrook, LI COMPUTER GENERAL STORE
Melville, FUTURE VISIONS COMPUTER STORE
Middletown, COMPUTER EMPORIUM
New Hartford, UPSTATE COMPUTER SHOP
New Hyde Park, BERLINER COMPUTER CENTER
New York, ARISTO CRAFT D. M
New York, COMPUTER ERA
New York, COMPUTER FACTORY
New York, COMPUTERLAND OF NYC
New York, DIGIBYTE SYSTEMS
New York, VIDEO CENTRAL
New York, WORLDWIDE MEDIA SERVICE
Patchogue, COMPUTER SHOPPE
Pelham, MEIZNER BUSINESS MACHINE
Plattsburg, PNR CLEAN AIR ENTERPRISES
Pleasantville, SUNBURST
Pleasantville, THE MICRO CENTER
Port Chester, A WORLD OF COMPUTERS
Poughkeepsie, AND OFFICE SYSTEMS
Red Hook, AMERICAN MICRO MEDIA
Rensselaer, SOFTRON SYSTEMS
Rochester, COMPUTER STORE
Rochester, HOME COMPUTER CENTER
Snyder, MODERN TEK SHOPS, INC
Spring Valley, COMPUTER STRATEGIES
Staten Island, C. HABILD OF NEW DORP
Staten Island, COMTECK
Syosset, SYOSSET VIDEO ELECTRONIC
Syracuse, PHANTOS RESEARCH
Troy, TREE OF KNOWLEDGE
Wappingers Falls, COMPUTER SYSTEMS SPECIALTY
White Plains, COMPUTER CORNER
White Plains, COMPUTERLAND
Williamsville, COMPUTER RESOURCES
NORTH CAROLINA
Aberdeen, KING ELECTRONICS
Asheville, HART, INC
Burlington, CAROLINA BIOLOGICAL SUPPLY
Elizabeth City, DATA SYSTEMS
Goldsboro, VIDEO CONNECTION/GOLDSBORO
Greensboro, BYTE SHOP
Havelock, SOUND MILL
Raleigh, ALPHA TECHNOLOGY, INC
Rocky Mount, SOUTHERN SOFTWARE CO.
Sylva, RAYBURN ELECTRONICS
Winston-Salem, ELECTRONIC ACCESSORIES
Winston-Salem, K & S NEWS STAND
OHIO
Akron, BASIC COMPUTER SHOP
Akron, MICRO CONCEPTS
Cincinnati, CINCINNATI COMPUTER STORE
Cincinnati, TWENTY-FIRST CENTURY SHOP
Cincinnati, WANNA PLAY
Cleveland, DIGITRENDS
Cleveland, JERRY'S COMPUTERS
Cleveland, THE ARCADE
Cleveland Heights, RESOURCES, INC.
Columbus, COMPUSERVE
Columbus, COMPUTERLAND
Columbus, MICRO ELECTRONICS
Columbus, MICROAGE COMPUTER STORE
Dayton, ALTAIR SYSTEMS, INC
Dayton, MICRO COMPUTER CENTER
Lancaster, ASTRO VIDEO ELECTRONICS, INC
Mayfield Heights, COMPUTERLAND
Meddlefield, JOBAR ENTERPRISES
North Olmsted, COMPUTERLAND
NW Philadelphia, LITTLE PROFESSOR BOOK CENTER
Reynoldsburg, UNIVERSAL AMATEUR RADIO, INC.
Toledo, COMPUTER STORE
Warren, COMPUTERLAND
OKLAHOMA
Guymon, RADIO SHACK
Moore, PROFILE MANAGEMENT
Sapulpa, VERN STREET PRODUCTS
Tulsa, COMPUTER STORE, INC.
Tulsa, COMPUTER WORLD
Tulsa, MICROCOMP SERVICES
Watonga, SOUNDS, ETCETRA
OREGON
Bandon, ACETEC COMPUTER SALES
Grants Pass, L & B ELECTRONICS
Portland, THE NOR'WESTER BOOKSHOP
Salem, COMPUTER SPECIALTIES
Sandy, PIONEER ELECTRONICS (R/S)
Tigard, COMPUTERLAND OF PORTLAND

PENNSYLVANIA
Altoona, J & E COMMUNICATIONS
Ardmore, ARDMORE HOBBIES, INC
Duncansville, J & E COMMUNICATIONS
Edinboro, BELL ELECTRONICS
Erie, ERIE COMPUTER
Fairview, BELL ELECTRONICS
Feasterville, OMNIFAX
Frazer, HEATHKIT ELECTRONIC CENTER
Gibsonia, COMPUTERLAND/PITTSBURGH
Honesdale, THE COMPUTER ROOM
Horsham, MIGHTY BYTE COMPUTER CENTER
Johnstown, MAFEX ASSOCIATES
Kingston, ARTCO ELECTRONICS
Lancaster, ALPHA COMPUTER CENTER
Lancaster, COMPUTER, TV AND SOUND
Latrobe, ROUTE 30 ELECTRONICS (R/S)
Malvern, NEVER LIMITED
Mechanicsburg, COMPUTERLAND OF HARRISBURG
Monroeville, COMPUTER WORKSHOPPE
New Brighton, TELEVISION PARTS CO.
New Hope, RUMPLESTILSKIN TOY SHOP
Philadelphia, OMNIFAX
Phoenixville, STEVENS RADIO SHACK
Pittsburgh, PITTSBURGH COMPUTER STORE
Pittsburgh, THE COMPUTER HOUSE
Ruffsdale, SPECIALTY GRAPHICS
Waynesburg, WAYNESBURG RADIO
Wexford, NORTH HILLS COMPUTER
Whitehall, COMPUTERLAND/LEHIGH VALLEY
RHODE ISLAND
Foster, COLONIAL ENTERPRISES, INC
Westerly, VIDEO ENCOUNTERS
SOUTH CAROLINA
Charleston, OMNI ELECTRONICS
Cordova, BROWNING ENTERPRISES
Greenville, COMPUTER DYNAMICS
SOUTH DAKOTA
Rapid City, COMPUTERLAND OF SOUTH DAKOTA
TENNESSEE
Chattanooga, CHATTANOOGA COMPUTER CENTER
Chattanooga, CHATTANOOGA MAGIC & FUN
Clarksville, ACE MINI SYSTEMS
Eagleville, SOFT-LINK PEOPLE
Harriman, WEBBS PHARMACY & ELECTRONIC
Memphis, COMPUTERLAB
Nashville, COMPUTER WORLD
Nashville, JAVAN ENTERPRISES
Tullahoma, H & H ELECTRONICS
TEXAS
Abilene, THE HOBBY CENTER
Aransas Pass, ROY'S CB & ELECTRONICS
Arlington, CODEDATA, INC
Arlington, COMPUTER PORT
Austin, COMPUTERS 'N THINGS
Austin, COMPUTERS TO GO
Beaumont, COMPUTER CONCEPTS
Beaumont, THE TOTAL COMPUTER STORE
Bellaire, COMPU SHOP
Breckenridge, RADIO 1000, INC
Dallas, THE PRINT OUT
El Paso, COMPUTER TECHNOLOGY
Fairfield, MACK'S TV
Fort Worth, CORSAIR
Fort Worth, STAR WARE
Houston, COMPU SHOP
Houston, COMPUTERLAND
Houston, GATEWAY ELECTRONICS
Houston, MICRO SOLUTIONS
Houston, TEXAS SOFTWARE
Houston, WAGHALTER BOOKS, INC.
Hurst, MICROAGE COMPUTER STORE
Katy, MARYMAC INDUSTRIES (R/S)
Lake Jackson, COMPUTERS BY O'NEILL
Mission, PANAMERICAN ELECTRONICS (R/S)
Missouri City, U & R ASSOCIATES INC.
Richardson, COMPUSHOP
San Angelo, COMPUTERS WEST
San Antonio, 80 SOFTWARE STORE
San Antonio, COMPUTER SOLUTIONS
San Antonio, MICRO MART
San Antonio, R.L. COLE'S ELECTRONICS
Temple, PROFESSIONAL COMPUTER ASSOCIATION
Webster, COMPUTEX
UTAH
Salt Lake City, COMPUTERLAND
Salt Lake City, FOOTHILL MODELERS
Salt Lake City, LIZCON TRADING
Salt Lake City, QUALITY TECHNOLOGY

continued

High-Res Alphanumerics

by Scott Norman

Produce unique character sets in high resolution on a TRS-80 Color Computer with 16K.

The Color Computer's most attractive feature is the high-resolution graphics capability of the 16K RAM/Extended Color Basic combination. I often plot mathematical functions using PMODE 4, which gives 256 by 192 addressable points on the video display. No built-in way exists to display text in any of the high-resolution modes. Since graphs need titles and labeled axes, I developed the program modules described in this article. Many high-resolution game displays can benefit from the on-screen scoring or alphanumeric prompts these modules provide.

Extended Color Basic's versatile Draw command defines my character set. Individual characters that are now string variables can be concatenated into labels by the execute (X) action option of the Draw command. Relative addressing of each character, plus an initial Blank and Move command, allow you to put the label wherever you like.

Here's an elementary example. Suppose you've defined the uppercase characters A, B, and C as string variables A$, B$, C$. Program line DRAW "BM 50,100; XA$; XB$; XC$;" writes the label ABC on a high-resolution graphics display, starting at the point X = 50, Y = 100.

Program Listing 1 contains the set of uppercase alphabetic characters, numerals, and miscellany I used. I constituted most of the characters in a 5 by 7 resolution element box, but since this is a custom set I've taken a few liberties—M and W are wider, for instance. I also put a proper descender on Q and an upper serif on I, but I encourage anyone using this technique to experiment to generate a pleasing display.

Users of Extended Color Basic will recognize the character definition syntax. If you are unfamiliar with the language, a segment like BM + 5, −5... is a relative addressing command. It means, "Starting from the present screen position, blank the CRT's electron beam and move five units to the right and five units up." You can then

Program Listing 1

```
 2 AA$="BM+1,ØU4E2R1F2D4BL3BU2R2BD2BR3"
 3 BB$="BM+2,ØU6BL1R3F1D1G1L1R1F1D1G1L3BR7"
 4 CC$="BM+5,-5H1L2G1D4F1R2E1BD1BR3"
 5 DD$="BM+1,ØU6R2F2D2G2L3BR7"
 6 EE$="BM+5,ØL4U6R4BD3BL2L1BD3BR6"
 7 FF$="BM+1,ØU6R4BD3BL2L1BD3BR6"
 8 GG$="BM+5,-5H1L2G1D4F1R3U2L1R3BR3BD2"
 9 HH$="BM+1,ØU6BR4D6BL3BU3R3BR4BD3"
1Ø II$="BM+5,-6L4R2D6L2R4BR3"
11 JJ$="BM+1,-2D1F1R2E1U5BD6BR3"
12 KK$="BM+1,ØU6D3R1E3G3F3BR3"
13 LL$="BM+1,-6D6R4BR3"
14 MM$="BM+1,ØU6R1F2D1U1E2R1D6BR3"
15 NN$="BM+1,ØU6R1D1F4D1R1U6D6BR3"
16 OO$="BM+5,-1U4H1L2G1D4F1R2BR4"
17 PP$="BM+1,ØU6R3F1D1G1L2BR6BD3"
18 QQ$="BM+5,-1U4H1L2G1D4F1R2F2BE2"
```

Listing 1 continues

The Key Box

Extended Color Basic
Color Computer
16K RAM

Listing 1 continued

```
19 RR$="BM+1,ØU6R3F1D1G1L2R1D1F2BR3"
2Ø SS$="BM+1,-1F1R2E1H4E1R2F1BD5BR3"
21 TT$="BM+3,ØU6L3R6BR2BD6"
22 UU$="BM+1,-6D5F1R2E1U5BD6BR3"
23 VV$="BM+1,-6D4F2E2U4BD6BR3"
24 WW$="BM+1,-6D6R1E2U1D1F2R1U6D6BR3"
25 XX$="BM+1,ØU1E4U1BL4D1F4D1BR3"
26 YY$="BM+1,-6D1F2D3U3E2U1BD6BR2"
27 ZZ$="BM+1,-6R4D1G4D1R4BR3"
28 BK$="BM+1,ØBR7"
29 PT$="BM+2,ØR1BR2"
3Ø NR$(1)="BM+2,-4E2D6BR4"
31 NR$(2)="BM+1,-5E1R2F1D1L1G3D1R4BR3"
32 NR$(3)="BM+1,-1F1R2E1U1H1E1U1H1L2G1BD5BR7"
33 NR$(4)="BM+4,ØU6L1G2D1R5BD3BR3"
34 NR$(5)="BM+1,-1F1R2E1U1H1L3U3R5BR3BD6"
35 NR$(6)="BM+5,-6L3G1D4F1R2E1U1H1L2BR6BD3"
36 NR$(7)="BM+1,-6R5D2G4BR7"
37 NR$(8)="BM+1,-1U1E1R2E1U1H1L2G1D1F1R2F1D1G1L2BR6"
38 NR$(9)="BM+1,ØR3E1U4H1L2G1D1F1R2BD3BR4"
39 NR$(Ø)="BM+6,-1U4H1L2G1D4F1R2BR4"
4Ø MI$="BM+2,-3R3BD3BR2"
```

```
  2 )
  . )
  . )  BASIC Statements of Program Listing 1
 4Ø )

1ØØ PMODE 4,1: PCLS: SCREEN 1,1
1Ø5 LINE (128,Ø)-(128,191),PSET
11Ø LINE (Ø,96)-(255,96),PSET
115 FOR P=8 TO 248 STEP 6Ø
12Ø LINE (P,94)-(P,98),PSET
125 NEXT
13Ø FOR P=46 TO 146 STEP 25
135 LINE (126,P)-(13Ø,P),PSET
14Ø NEXT
141 DRAW "BM4,8;XTT$;XRR$;XII$;XGG$;XOO$;XNN$;XOO$;
    XMM$;XEE$;XTT$;XRR$;XII$;XCC$;"
142 DRAW "BM5,2Ø;XFF$;XUU$;XNN$;XCC$;XTT$;XII$;
    XOO$;XNN$;XSS$;"
15Ø FOR X=-18Ø TO 18Ø STEP 5
155 TH=X/57.3: XS=128+.67*X
16Ø F1=5Ø*SIN(TH)
165 F2=5Ø*COS(TH)
17Ø PSET(XS,96-F1,1)
175 PSET(XS,96-F2,1)
18Ø NEXT
185 DRAW "BM188,44;XSS$;XII$;XNN$;XEE$;"
19Ø DRAW "BM188,154;XCC$;XOO$;XSS$;XII$;XNN$;XEE$;"
195 DRAW "BM174,1Ø6;XNR$(9);XNR$(Ø);"
2ØØ DRAW "BM228,1Ø6;XNR$(1);XNR$(8);XNR$(Ø);"
2Ø5 DRAW "BM2,1Ø6;XMI$;XNR$(1);XNR$(8);XNR$(Ø);"
21Ø DRAW "BM62,1Ø6;XMI$;XNR$(9);XNR$(Ø);"
215 DRAW "BM116,49;XNR$(1);"
22Ø DRAW "BM11Ø,149;XMI$;XNR$(1);"
225 GOTO 225
```

Program Listing 2

draw a character. In the context of these statements, L, R, U, and D are commands for left, right, up, and down, while E, F, G, and H denote 45-degree motions to the northeast, southeast, and so on. Character definitions end with blanked-out motions like BR6, "blank and move six units to the right," to set you up to specify the next character in a string.

You can call the characters anything you please. I denoted the letters as AA$ and so on in order to reserve one-letter string variable names like A$ for general use in my programs. I also included a callable blank BK$, a decimal point PT$, and a hyphen MI$. I defined the numerals as elements of a 10-element string variable array NR$(I) to call them out to construct numeric labels computed within a program rather than being specified at the beginning.

Applications

Unlike data statements, these character definitions must appear in a program before the statements in which they are called. For that reason, I gave the definitions low statement numbers (2–40). If your application requires a large amount of memory for extra video pages, use line 1 to avoid wiping out any of the character definitions. Set up the high-resolution PMODE 4 when you begin plotting.

Program Listing 2 is an application in which the positions and definitions of the alphanumerics are known at the outset and can be specified by the programmer. It plots the sine and cosine functions over one cycle, with a label for each curve, a title for the whole display, and values for a few major points on each axis. Line 100 sets the high-resolution mode, lines 105–140 draw the axes and tic marks, and lines 141–142 write the title Trigonometric Functions in the upper left corner of the screen. The execute action option strings together individual graphics symbols by prefixing their names with X and postfixing with a semicolon. Lines 150–180 perform the actual computations and function plotting (by the slow PSET process in this case). The rest of the program labels the curves and puts numerical values on the tic marks.

The full syntax of the Draw command permits rotating a predefined set of operations in 90-degree increments. In particular, prefixing A3 as in DRAW "A3BM..." results in a character string written vertically upward. You might want to label the vertical axis of a graph this way if you don't want to cock your head 90 degrees to

the left in order to read it! If you use this option, remove it with a prefixed A0 for the next character string that you want to read normally.

If you don't know at the outset what the scale of a graphical axis will be, how can you plot the values calculated within the program? Program Listing 3 demonstrates a fairly simple approach, using the string-variable array. This program calculates the ratio of two numbers entered from the keyboard and displays the quotient in high-resolution mode. Line 65 converts the quotient, Q, to a string and establishes a buffer array BF(I). The elements of BF(I) are the digits of Q. If a particular digit is a decimal point (ASCII code 46), however, then the value 46 is stored in the corresponding element of BF(I) by lines 80–95. Line 100 shifts the display from the low-resolution Color Basic mode in which you entered the numerator and denominator to high-resolution for the output. A starting position 100 units down the left side of the display is established, any leading zero suppressed (line 110), and the elements of BF(I) interrogated one at a time. A value of 46 for any element draws the decimal point; any other value, 0–9, calls up the corresponding high-resolution graphics character. A simplified form of the Draw command concatenates strings with a For...Next loop: DRAW "XNR$ (BF(I));" where you enter only the closing semicolon after you supply an originating position for the string, as in line 105.

Writers of game programs can apply this technique of dissection and reconstruction of numerical strings to alphabetic characters if there is reason to compute alphabetic outputs based on the course of action.

Modifications

You can experiment with the character set defined in Listing 1 and invent your own sets, tailored to your own interests. You can add lowercase letters, Greek letters, and other mathematical symbols. You can add electronic symbols and the common representations of digital logic elements. You might write a Color Computer program to convert a logical expression into a block diagram for electronic realization. Memory usage will limit the imagination you can exercise; for the record, the character set definitions of Listing 1 occupy about 1,700 bytes. ■

Scott Norman can be reached at 8 Doris Road, Framingham, MA 01701.

```
 2 )
 . )
 . )   BASIC Statements of Program Listing 1
4Ø )

45 CLS: PRINT "CALCULATE & HI-RES PRINT": PRINT
5Ø INPUT "NUMERATOR";N
55 PRINT: INPUT "DENOMINATOR";D
6Ø Q=N/D
65 N$=STR$(Q)
7Ø L=LEN(N$)
75 DIM BF(L)
8Ø FOR I=1 TO L
85 A=ASC(MID$(N$,I,1))
9Ø IF A<>46 THEN BF(I)=VAL(MID$(N$,I,1)) ELSE BF(I)=46
95 NEXT
1ØØ PMODE 4,1: PCLS: SCREEN 1,1
1Ø5 DRAW "BMØ,1ØØ"
11Ø IF BF(1)=Ø THEN 12Ø
115 DRAW "XNR$(BF(1));"
12Ø FOR I=2 TO L
125 IF BF(I)=46 THEN DRAW "XPT$;" ELSE 135
13Ø GOTO 14Ø
135 DRAW "XNR$(BF(I));"
14Ø NEXT
145 GOTO 145
```

Program Listing 3

✓See List of Advertisers on Page 563

Teach Your Computer to Talk

by Richard Seymour

What will your Color Computer's first words be? Here's a step-by-step method in Assembly language to get you and your CC speaking.

The joy of hearing your computer's first words is second only to that of seeing your first program write something intelligent on the terminal.

I will describe a program I use on my Color Computer, but you can adapt the method to any computer. The computer converts a short amount of speech into a stream of numbers. This is called digitization. When you replay these numbers they recreate sound close to the original. Once you have recorded the original sound in the computer's memory, you can manipulate it for analysis, special effects or to minimize the space required.

The Key Box

Color Computer
Color Basic
Extended Color Basic
16K RAM
Cassette recorder

The Color Computer contains the hardware necessary for the conversion method I chose. Radio Shack's Skiing game cartridge saying "Get ready, get set" suggested this approach. I will also describe the computer's other digitization possibilities.

Theory

Any sound fed into a microphone consists of a complex electrical signal. Figure 1 shows an example of such a signal. Classical digitizing methods slice this signal into equal time intervals (Fig. 1b). At the end of each slice the voltage of the signal is measured and stored. The sound is recreated by reproducing those levels at the same intervals.

Two factors affect the sound quality. The first is how well the system can reproduce the tones (frequencies) in the original sound. This is called frequency response. Generally, you can reproduce the original frequencies to any desired accuracy by adjusting the time interval. Shorter intervals give better reproduction (see Figs. 1c and 1d). A rule of thumb is that you must measure twice as fast as the highest frequency you want to reproduce. Short time slices require recording more measurements.

The second factor is the resolution of the signal's voltage level (amplitude or volume). In a digital system the number of steps available between the lowest and highest voltages measured or reproduced describe the voltage

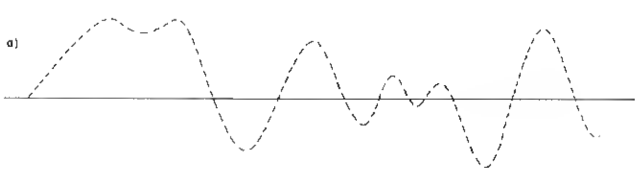


Figure 1a

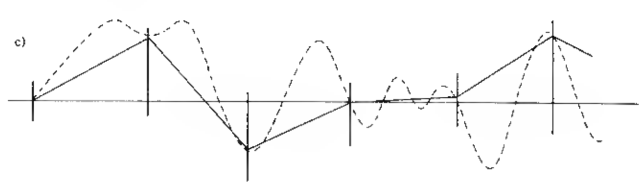


Figure 1c

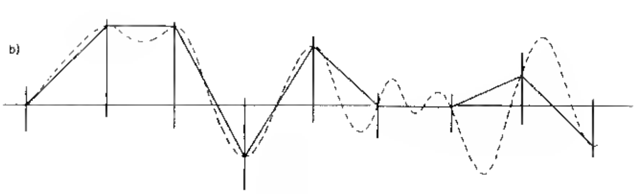


Figure 1b

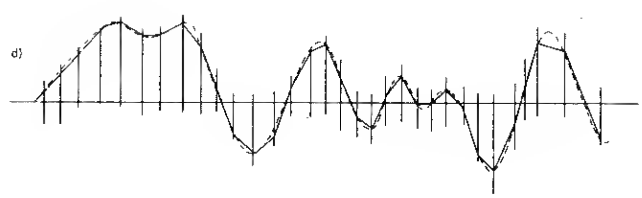


Figure 1d

level. This is normally expressed by the number of data bits required for that number of levels. Therefore, a 6-bit system can provide 64 volume levels. This same number describes the hardware that converts the voltage to a number. Such hardware is called an analog-to-digital converter (ADC) for creating numbers, or a digital-to-analog converter (DAC) for recreating voltages.

The Color Computer's two ways of analyzing incoming voltages are based on the same principle—comparison to an internal reference voltage. One serves the cassette tape input, and compares the signal to zero volts. The other compares joystick input to the output of a 6-bit (64-level) DAC directed by the computer. The computer performs the A/D function by trying different voltages until it finds the closest match to the input voltage.

Why are there two ways? Radio Shack records information on cassette with two distinct tones. This allows recovery by simply measuring the time between the zero-crossings of the recorder's signal. Binary zeros are marked by crossings spaced twice as far apart as binary ones. This is done very quickly. The 6-bit method, which provides much more information about the input signal, requires more time to measure. When digitizing a rapidly changing signal such as voice or music, that time sets the upper limit on which frequencies can be reliably detected.

My hardware for this project included a 16K Extended Color Basic Color Computer, a 1967 GE Porta-Color tv and a $25 GE cassette recorder. Just as a $1,000 stereo receiver would be wasted on $25 speakers, the quality of the audio path available did not inspire me to attempt ultra high fidelity. I also wanted fast reward for

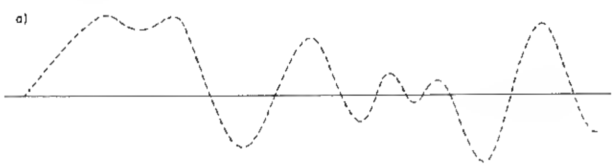


Figure 2a

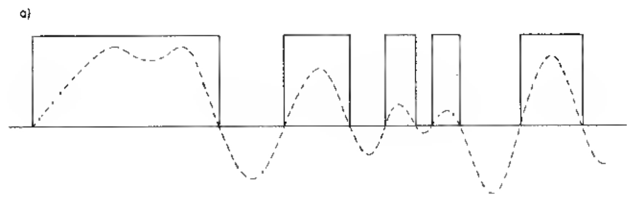


Figure 3a

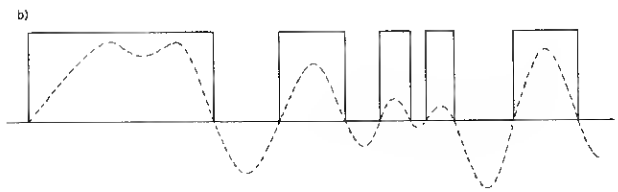


Figure 2b

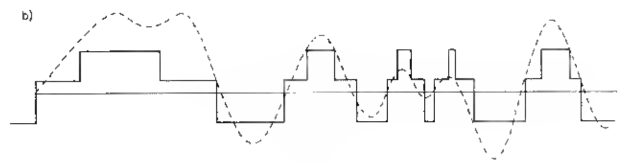


Figure 3b

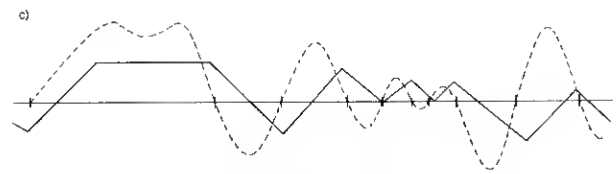


Figure 2c

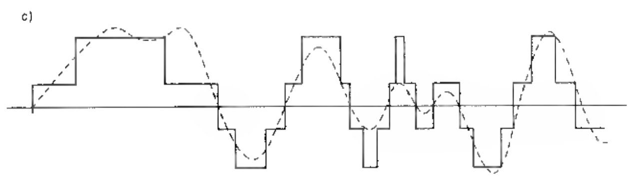


Figure 3c

my efforts. So I chose to duplicate the cassette tape's input method for its programming simplicity.

Since I was only recording the duration between zero crossings, I decided first to simply send on/off signals back out to the tv set. The computer's single-bit sound system did not work reliably. The 6-bit D/A did work, so I just hit it with a zero or 24 value. Other experiments showed that the absolute values did not matter—only the difference between them. Therefore, the value 24 minus zero sets the volume heard over the speaker. The same result is produced with 30 and 54. I chose 24 by running a joystick-controlled noise program until I found an acceptable value. I encourage you to experiment with these values. Remember to multiply the value you choose by four (shift up 2 bits) to match it to the pins the Color Computer uses for the DAC's input.

Fig. 2 shows the liberties my conversion scheme wreaks upon a sample signal. The original signal (2a) becomes the time blocks (2b) which the computer-to-speaker rounds off to 2c.

Because I use cassette input, I found it easiest to pre-record sample speech. I can exactly reproduce experiments in memory-reduction techniques. Most cassette recorders pass what the microphone hears through the earphone connector while recording, so I use the recorder's microphone to speak directly to the computer. Unplug the computer-to-cassette line from the microphone or auxiliary jack. Leave the earphone line connected. Set the recorder to record and speak into the microphone. You get better quality sound if you use a separate microphone; the one in the recorder picks up a lot of motor noise. The system consists of the tape recorder playing into the normal cassette input, with the reproduced sound playing back through the television speaker.

The program includes three sections: Record, Playback and Basic (which places the other two into memory and interacts with the operator).

Type in the program in Program Listing 1. You may delete all comments in lowercase to save typing. Program Listing 2 is the same program in regular Basic without comments. Program Listing 3 is an Assembly-language listing for the data statements of the Basic programs.

Bookkeeping

For Extended Basic, I put the recorded numbers into the graphic screen area. I watch the data as it comes in and receive a fascinating view of the effect of pauses and noise on the amount of memory used. For regular Basic, I put the storage above the Basic code, behind the cleared area. I put the machine-language part at the top of memory, using Basic's Clear statement to avoid stepping on it with other parts of the program.

One second of speech takes about 2K bytes of memory. Use that number to plan how much memory you want to give to speech storage. I put the table addresses in separate Basic lines to make them easier to find and change.

Recording

The Recording section of the program has four sections. Initialization sets the computer's registers. The "loop while above zero" section times the positive half of the signals. The "loop while below zero" section times the negative half of every wave. The fourth section checks to see that the program has not run out of storage space.

The Color Computer's heart is a Motorola 6809. It has a variety of internal memories (registers) that you can use to count items or point to other memory locations or both. In this case, I tell the control register to ignore any other disturbances from the outside world (interrupts). This is necessary since I am trying to get accurate timing information by counting the number of times a loop is executed. It would not do to have the computer off somewhere else while my tape played on.

The Color Computer presents the status (zero or one) of the tape input detector as the bottom bit (least significant) of memory location 65312 (FF20). I load the U register with FF20. I load the A register with a one, which I will compare against the status bit to see if it is on or off. Then I load the X register with the starting address of the number storage area. That completes the initialization section.

The program then drops into the "watch while high" section. First it clears the B register, which will serve as the loop counter. Then it adds one to the B register and tests the status bit at FF20. If the bit is still on, the program loops back to the add-one-to-B register instruction. Once the status bit becomes zero, the program falls past the branch-if-not-equal instruction. There it stores the B register count at the address in the X register. That same instruction adds one to the X register, pointing it to the next table location.

```
1 ' TALK  for Extended Color Basic
2 ' (c) Richard Seymour aug 1981
5 ' reserve memory and set up entry addresses for machine language
10 CLEAR 200,16200 : MOTOR ON : AUDIO ON : REC=16200 : PLY=16231
15 '  tell extended basic about them, copy the data to the area
20 DEFUSR0=REC : DEFUSR1=PLY : LOC=REC : GOSUB 9900
25 '  tell people how to play
30 PRINT"HOLD RIGHT 'FIRE' BUTTON"
35 PRINT" TO RECORD ANOTHER PIECE"
40 PRINT"HOLD 'BREAK' TO EXIT PROGRAM"
45 '  turn on the sound and wait for a "return"
50 AUDIO ON : INPUT "HIT RETURN TO START CONVERSION";A$
55 '  go to high resolution graphics and select a graphics display
60 PMODE 4,1 : SCREEN 1,I
65 '  perform the "record" subroutine
70 AUDIO ON : A=USR0(0)
75 '  a moment of silence, please
80 FOR I=1 TO 200 : NEXT I
85 '  "Go ahead, Sam.  Play it."
90 A=USR1(0)
95 '  look for a struck key or joystick button, rerecord if seen
100 IF (PEEK(&HFF00) AND 1)=0 THEN 50
105 '  just play it again if no buttons were pushed
110 GOTO 80
115 '
9000 '   Record portion of program
9005 DATA 1A, 50, CE, FF, 20, 8E
9009 'table start address follows
9010 DATA 06, 00
9015 '  next line starts at 16208
9020 DATA 86, 01, 5F, 5C, A5, C4, 26, FB, E7, 80
9025 '  next line starts at 16218
9030 DATA 5F, 5C, A5, C4, 27, FB, E7, 80, 8C
9034 'table end address follows
9035 DATA 1D, FC
9040 DATA 25, EB
9045 '  Play starts here (16231)
9050 DATA 1A, 50,  CE, FF, 20, 6F, 43
9055 '  next line starts at 16238
9060 DATA CC, F8, 3C, A7, 42, E7, 43, CC, B4, 35, A7, 5D, E7, 5F
9065 '  next line starts at 16252
9069 'table start address follows
9070 DATA 8E, 06, 00
9075 '  next line starts at 16255
9079 'next 60=volume
9080 DATA 86, 60
9085 '  next line starts at 16257
9090 DATA E6, 80, A7, C4, 5A, 26, FB, 4F, E6, 80
9095 '  next line starts at 16267
9100 DATA A7, C4, 5A, 26, FB, 8C
9104 'table end address follows
9105 DATA 1D, FC
9110 '  next line starts at 16275
9115 DATA 25, EA
9120 '  next line starts at 16277
9125 DATA 39 : '   END of subroutine
9900 DATA 1000 : '  flag for end of transfer
9905 '  read value, convert to variable
9910 READ A$ : A=VAL("&H"+A$)
9915 '  we're finished if it's over 255 (FF hex)
9920 IF A>255 THEN RETURN
9925 '  otherwise, put it into memory, increment LOC, and get another
9930 POKE LOC,A : LOC=LOC+1 : GOTO 9910
9998 '
9999 END : ' of everything
```

Program Listing 1. Extended Basic Version

Next the program drops into the "watch while low" section. This is just like the "while high" section, except it loops while the bit is zero. Once the bit goes non-zero again, the program moves to another "store B at X" instruction. After that the program checks that X has not risen to the top end of memory space reserved for storage. If X is still below that limit, the program branches back up to the "while high" loop. If X reaches its limit, the program moves from the check section into the Playback program. You could provide an immediate Return here, if you do not want an instant replay.

Playback

You can enter the Playback program directly from Basic, so start by reinitializing any registers you need. Again, the U register points to the hardware control locations in memory. Clear the hardware register controlling the sound source selection to provide access to its direction control register. Then set this to guarantee your choice of the 6-bit DAC. Reset the control register to its normal state, with the sound output enabled. Setting FF1D and FF1F selects the path from the DAC to tv speaker. Finally, the X register points to the beginning of the sound table.

As the first step of the "send high value" loop, the A register is set with the volume level selected. This is only done at the start of a "send high" section, not within the fast inner loop.

Then the program loads the B register with the recorded loop count fetched from the table (pointed to by the X register, which is incremented after the fetch). The A volume value is sent to the DAC, located at FF20. Subtract one from the B register count. If it has not reached zero, branch back to the send A instruction. When it does reach zero, we proceed to the "send low" section.

As a first step, clear the volume (A) register to zero. Then fill B with the next count from the table. Send A to FF20, decrement B, and check for zero. Loop to the local send A instruction until B goes to zero. Then check that the X register is still within the table's limits. If it is, branch back up to start the "send high" section again. When X reaches or exceeds its limit, the program returns to Basic.

All of the innermost loops in record and playback take exactly the same time—nine 6809 cycles, or a little more than 11 microseconds in the Color Computer.

Basic

The Extended Basic program reserves the memory above 16200 for use by the machine-language code. I turn on the tape recorder's motor; the tape-to-speaker sound path allows me to hear the tape before I start the conversion. I set the starting addresses of the record and playback routines into Extended Basic's user-function table. Regular Basic and Extended Basic are incompatible here. In regular Basic, you POKE the starting address of the single USR function to locations 275 and 276 (decimal). Extended Basic does not allow that. Extended Basic requires the DEFUSR0 = ADDRESS convention. A scratch variable LOC points to the location to receive the next byte of machine-language code. I then go to the move-machine-language subroutine.

Lines 9900–9930 read string data values as hexadecimal values, and POKE them into the assigned area. Using hexadecimal data values allows direct use of Motorola's hex reference cards. It works by converting all strings

(such as A9) to the hexadecimal constant form (&HA9) recognized by Extended Basic. This form handled by the VAL function generates the decimal value handed over by POKE. Temporary variable A detects the end of the table. This also catches many missed-comma typing errors in the data list. When the over-255 value planted in 9900 is detected, the machine-language load is stopped, and control returns to the main program section.

The regular Basic version of the program has the same data except for the table addresses. For simplicity, I present the data as decimal numbers.

Now we turn on the tape-to-tv path again, because we can enter statement 50 from below as well (that's line 60 in regular Basic). The programs pause for keyboard input. In Extended Basic the computer automatically goes to Text mode when it hits the Input command. For regular Basic I force the video system back into line with the GOSUB 300 section.

The input statement allows final positioning of the source tape before starting digitization. When you hit return the computer flips to maximum graphics mode to show 6K bytes on the screen. GOSUB 200 in regular Basic does this. Then the Record routine starts.

As the recording progresses, you will see short horizontal lines and rows of dots written or modified on the graphics screen. Each group of eight dots represents one byte of the table. They group into 32 columns. The rightmost bit of each column is the one bit of that count byte. A tone of short counts (or tape hiss) tends to form vertical lines along their zone's right edge. You will see the bytes change as a ripple across each of the 192 lines, from left to right, from top to bottom. When the ripple hits the bottom, the recording section stops, and Playback begins. The computer speaks!

After one pass through Playback, control returns to Basic. At this point I use a loop to cause a slight silent delay, and then enter the Playback routine directly. The computer speaks again!

I then check the fire button on the right joystick. This check can also be satisfied by holding down any key between A and G. If the key (or button) is not hit, I repeat the Playback loop. If the key (or button) is down, I go back for another Record session.

To stop the program, Extended Basic responds to the break key. Regular Basic needs to be told how to restore the text screen, so the program checks for the left joystick fire button, too. If that button is pushed, GOSUB 300 resets the screen before the program stops. If you break out of the program, you may type GOSUB 300 (even if you can't see it) to get the text screen back. The alternative is to push the reset button.

More and Better

What next? I have tried to lower storage requirements by increasing the loop times. This means each count in the table covers a wider time slice. Higher frequencies will not be seen. Since higher frequencies make lots of table entries in a short time, this trick saves a good deal of memory. I added sections to scan through the data and show graphs of how many entries (channels) contain which counts. This is a rough form of spectrum analyzer.

I have also explored removing or replacing ranges of table entries. If I replace all entries below 16 with one, the speech becomes garbage. But if I replace those same entries with random numbers between one and 16, the speech is almost unchanged.

Loss of the high frequencies affects some words before others. For example, "cold mysterious," when robbed of its high frequencies by cutting below 16 or by loop-lengthening, becomes "cold whoosh." Perhaps you could generate a chart of low-frequency words (like "word") that can be stored efficiently. If the original sound is loud and full of low frequencies, the 6K region can hold more than 10 seconds of speech. Loudness keeps the smaller-level high frequencies from making zero crossings. The low frequencies make each table byte take longer. If you are now recording 500-cycle tones instead of 1,000-cycle tones, each byte holds the loop for twice as long. By having that repeated 6,000 times, the total time spent by the record/playback is more than doubled.

Another obvious step is to improve the volume resolution by using the A/D ability of the computer. This need not be done to the full 6-bit resolution if time is a problem. You can reach the proportional input through either of the joystick inputs.

```
1 '  TALK  for Color Computer Regular Basic
2 '   (c) Richard Seymour  aug 1981
10 CLEAR 200,8192 : MOTOR ON : AUDIO ON : REC=16200
20 R1=INT( REC/256 ) : R2= REC - R1*256 : PLY=R2 + 31
30 PRINT"HOLD THE RIGHT 'FIRE' BUTTON":PRINT" TO RECORD ANOTHER SECTION"
40 PRINT"HOLD THE LEFT 'FIRE' BUTTON":PRINT" TO EXIT THE PROGRAM"
50 POKE 275,R1 : POKE 276,R2 : LOC=REC : GOSUB 9900
60 GOSUB 300 : AUDIO ON : INPUT "HIT RETURN TO START CONVERSION";A$
70 GOSUB 200 : ' start graphics
80 POKE 276,R2 : A=USR(0)
90 FOR I=1 TO 200 : NEXT I
100 POKE 276,PLY : A=USR(0)
110 A=PEEK(65280) : IF (A AND 1)=0 THEN 60
120 IF (A AND 2)=0 THEN GOSUB 300 : END
130 GO TO 90
200 POKE 65314,249 : POKE 65475,1 : POKE 65477,1 : POKE 65480,1 : POKE 65487,1
210 RETURN
300 POKE 65314,0 : POKE 65474,1 : POKE 65476,1 : POKE 65481,1 : POKE 65486,1
310 RETURN
9000 DATA  26,  80, 206, 255,  32, 142
9005 'table start address follows
9010 DATA 32,0
9020 DATA 134,   1,  95,  92, 165, 196,  38, 251, 231, 128
9030 DATA  95,  92, 165, 196,  39, 251, 231, 128, 140
9035 'table end address follows
9040 DATA 55,252
9050 DATA  37, 235,  26,  80, 206, 255
9060 DATA  32, 111,  67, 204, 249,  60, 167
9070 DATA  66, 231,  67, 204, 180,  53, 167
9080 DATA  93, 231,  95, 142
9085 ' table start address follows
9090 DATA 32, 0
9095 ' the next 96 is the volume
9100 DATA 134, 96
9110 DATA 230, 128, 167, 196,  90,  38, 251
9120 DATA  79, 230, 128, 167, 196,  90,  38, 251, 140
9125 'table end address follows
9130 DATA 55, 252
9140 DATA  37, 234
9150 DATA  57
9900 DATA 1000 : ' flag to stop transfer
9910 READ A
9920 IF A>255 THEN RETURN
9930 POKE LOC,A : LOC=LOC+1 : GOTO 9910
9999 END
```

Program Listing 2. Regular Basic Version

Another area to explore is shaping the output to something smoother than on/off. You can do this by dividing the loop count into quarters. Then send a 12 for the first quarter, 24 for the middle two quarters, and finish with 12 for the last quarter. This would produce the wave in Fig. 3a. By shifting to 16 and 24 for the on cycle, and eight and zero for the off cycle, you should get even smoother sound.

The Color Computer uses a similar technique for the waveform it sends to the cassette tape, but it reads values from a table stepped through at a constant rate. To achieve the higher frequency it uses every other value from the table. That table can be found starting at location A85C (decimal 43100) in the regular Basic ROM.

Other Computers

This method requires a single input bit driven by your sound source. Like the Color Computer, you can use the cassette inputs of other computers to do this. If you can drive a peripheral input adapter (PIA) pin from a sound source, you could use it. You cannot connect a normal digital integrated circuit to an alternating signal. When the signal drops below −0.5 volts, it would damage the chip. ■

Richard Seymour (Univ. of Washington, Nuclear Physics Lab, GL-10, Seattle, WA 98195) has been doing hardware and software development and management for 15 years.

```
TALK Assembly language routine
Loc    Hex                Source                Comments
Dec    Code               Code

16200  1A 50     RECORD:  ORCC   #50      ; block FIRQ and IRQ interrupts
16202  CE FF 20           LDU    #FF20    ; address of cassette input flag
16205  8E 06 00           LDX    #0600    ; table starting address
16208  86 01              LDA    #01      ; one bit for cassette flag
16210  5F        HIGH:    CLRB            ; start loopcount at zero
16211  5C        HLOOP:   INCB            ; count the loop
16212  A5 C4              BITA   ,U       ; check 1 bit at FF20
16214  26 FB              BNE    HLOOP    ; loop it on
16216  E7 80              STB    ,X+      ; put count in table
16218  5F                 CLRB            ; reset count to zero
16219  5C        LOW:     INCB            ; count this loop
16220  A5 C4              BITA   ,U       ; check the cassette bit
16222  27 FB              BEQ    LOW      ; loop while off
16224  E7 80              STB    ,X+      ; put count in table
16226  8C 1D FC            CMPX   #1DFC    ; table end address
16229  25 EB              BLO    HIGH     ; keep recording if ok
16231  1A 50     PLAY:    ORCC   #50      ; block FIRQ and IRQ
16233  CE FF 20           LDU    #FF20    ; address of D/A output
16236  6F 43              CLR    3,U      ; clear FF23 register
16238  CC F8 3C           LDD    #F83C    ; pattern for 6-bit sound
16241  A7 42              STA    2,U      ; set the direction register
16243  E7 43              STB    3,U      ; set sound out enable
16245  CC B4 35           LDD    #B435    ; pattern for sound path
16248  A7 5D              STA    -3,U     ; set the analog switch
16250  E7 5F              STB    -1,U     ; at FF1D and FF1F
16252  8E 06 00           LDX    #0600    ; beginning of table
16255  86 60     HIOUT:   LDA    #60      ; volume value (times 4)
16257  E6 80              LDB    ,X+      ; get next loop count
16259  A7 C4     HILOOP:  STA    ,U       ; turn on D/A
16261  5A                 DECB            ; downcount B register
16262  26 FB              BNE    HILOOP   ; until zero
16264  4F                 CLRA            ; zero volume
16265  E6 80              LDB    ,X+      ; next table entry
16267  A7 C4     LOLOOP:  STA    ,U       ; turn off D/A
16269  5A                 DECB            ; down we go
16270  26 FB              BNE    LOLOOP   ; until run to zero
16272  8C 1D FC           CMPX   #1DFC    ; at top of table yet?
16275  25 EA              BLO    HIOUT    ; if not, do more
16277  39                 RTS             ; otherwise, return to Basic
```

Program Listing 3. Assembly Language Routine

Five Games for 4K

by James Wood

Don't despair if your Color Computer has only 4K memory—here are Pong, Bingo, Simon, and Hangman clones written to fit your system.

The 4K Color Computer is limited by its small memory, but programs can be developed for it. Here are five games you can play on a 4K or larger Color Computer.

Rocket displays the number of hits, number of rockets, and number of shots at the top of the screen. A rocket (actually a rectangle) moves across the screen at various heights. A base at the lower right corner fires a missile when you press the F key. When you hit the rocket, a graphic explosion results. Make up your own rules; find out how many rockets you can hit with 10 shots or how many of the first 20 rockets you can hit. Adjust the program to fit your rules.

```
5 'ROCKET
1Ø 'JAMES W. WOOD, 424 N. MISSOURI,ATWOOD, IL, 61913
2Ø CLSØ:PRINT@Ø,"":PRINT@4,"HITS";:PRINT@27,"SHOTS";:PRINT@13,"R
OCKETS";
3Ø POKE153Ø,143:POKE1532,143
4Ø Z$=CHR$(128)+CHR$(128)+CHR$(128)+CHR$(128)+CHR$(128)
5Ø F$=CHR$(128)+CHR$(175)+CHR$(175)+CHR$(175)
6Ø H=H+1:PRINT@8,H;:E=RND(13)*32+32:PRINT@E,F$;
7Ø F=E
8Ø PRINT@E,F$;
9Ø E$=INKEY$
1ØØ IFE$="F"THENV=V+1:PRINT@23,V;:GOTO12Ø
11Ø E=E+1:IFE>F+28THEN2ØØELSE8Ø
12Ø FORQ=5Ø7TO35STEP-32
13Ø PRINT@Q,CHR$(255);:PRINT@Q,CHR$(128);
14Ø E=E+1:IFE>F+28THEN2ØØ
15Ø PRINT@E,F$;
16Ø IFQ=E+1ORQ=E+2ORQ=E+3THEN17ØELSE18Ø
17Ø W=W+1:PRINT@Ø,W;:GOSUB21Ø:FORTT=1TO5Ø:NEXTTT:GOTO6Ø
18Ø NEXTQ
19Ø E=E+1:GOTO8Ø
2ØØ PRINT@E-1,Z$;:GOTO6Ø
21Ø FORWW=1TO5:SOUND18Ø,1:PRINT@Q-2,CHR$(RND(128)+127)+CHR$(RND(
128)+127)+CHR$(RND(128)+127)+CHR$(RND(128)+127)+CHR$(RND(128)+12
7);:NEXTWW:PRINT@Q-2,Z$;:RETURN
```

Program Listing 1

"Here are five games you can play on a 4K or larger Color Computer."

Pong is similar to the tv pong games. The paddles move continuously if you keep the arrow keys pressed.

For Bingo, you will need markers and bingo cards. The computer generates bingo calls and graphically displays them on a chart. Each number appears on the screen as orange colored but turns red as subsequent numbers are called, so you can easily find the last call or check a winning card.

Program Listing 2

```
5 'PONG
7 'JAMES W. WOOD, 424 N. MISSOURI, ATWOOD, IL, 61913
1Ø CLS:PRINT:PRINTTAB(13)"PONG":PRINT:PRINT"PLAYER ON LEFT":PRIN
T" USES UP ARROW AND DOWN ARROW":PRINT:PRINT"PLAYER ON RIGHT":P
RINT" RIGHT ARROW IS UP":PRINT" LEFT ARROW IS DOWN":FORTI=1TO3
ØØØ:NEXTTI
2Ø XX=RND(9)-1
3Ø Q=2Ø:A=25:L=25:O=2Ø:CLSØ
4Ø PRINT@66,LL;:PRINT@91,RR;
5Ø FORY=9TO3Ø:SET(Ø,Y,XX):SET(63,Y,XX):NEXTY
6Ø C=1:K=1:PRINT@76,"PONG";
7Ø FORJ=2ØTO25:SET(7,J,XX):SET(56,J,XX):NEXTJ
8Ø FORX=1TO62:SET(X,9,XX):SET(X,3Ø,XX):NEXTX
9Ø Y=1Ø:X=RND(4Ø)+1Ø
1ØØ SET(X,Y,XX)
11Ø FORJ=1TO2:NEXTJ:RESET(X,Y)
12Ø IFPEEK(344)=247THENGOSUB35Ø
13Ø IFPEEK(343)=247THENGOSUB37Ø
14Ø IFPEEK(341)=247THENGOSUB39Ø
15Ø IFPEEK(342)=247THENGOSUB41Ø
16Ø X=X+C
17Ø Y=Y+K
```

Listing 2 continues

The Key Box

Color Computer
4K RAM
Color Basic or Extended Color Basic

Listing 2 continued

```
180 IFY<30ANDY>9THEN220
190 IFX>58THEN300
200 IFX<6THEN280
210 IFPOINT(X,Y)THEN260
220 IFX>58THEN300
230 IFX<5THEN280
240 IFPOINT(X,Y)THEN270
250 GOTO100
260 SOUND100,1:K=-K:GOTO170
270 SOUND200,1:C=-C:GOTO160
280 SOUND20,5:RR=RR+1:PRINT@91,RR;:IFRR=15THEN320
290 GOTO90
300 SOUND20,5:LL=LL+1:PRINT@66,LL;:IFLL=15THEN320
310 GOTO90
320 PRINT@485,"PLAY AGAIN":KI$=INKEY$
330 B$=INKEY$:IFB$=""THEN330
340 IFB$="Y"THENRUN1ELSEIFB$="N"THENENDELSE330
350 IFO<11THENRETURN
360 RESET(56,L):O=O-1:L=L-1:SET(56,O,XX):RETURN
370 IFL>28THENRETURN
380 RESET(56,O):O=O+1:L=L+1:SET(56,L,XX):RETURN
390 IFQ<11THENRETURN
400 RESET(7,A):Q=Q-1:A=A-1:SET(7,Q,XX):RETURN
410 IFA>28THENRETURN
420 RESET(7,Q):Q=Q+1:A=A+1:SET(7,A,XX):RETURN
```

```
5 'BINGO
7 'JAMES W. WOOD, 424 N. MISSOURI, ATWOOD, IL, 61913
10 CLS:PRINT@200,"BINGO":PRINT:PRINT"   PRESS <ENTER> FOR NEXT C
ALL.":PRINT:PRINT"PRESS <Z> IF READY FOR NEXT GAME"
20 PRINT@420,"PRESS <ENTER> TO CONTINUE"
30 A$=INKEY$:IFA$=""THEN30
40 CLS0:PRINT@4,"B";:PRINT@9,"I";:PRINT@14,"N";:PRINT@19,"G";:PR
INT@24,"O";
50 FORC=1TO15:PRINT@32*C,C;:PRINT@32*C+5,C+15;:PRINT@32*C+10,C+3
0;:PRINT@32*C+15,C+45;:PRINT@32*C+20,C+60;:NEXTC
60 FORC=35TO291STEP32:PRINT@C,CHR$(143);:NEXTC
70 DIMB(75)
80 A=RND(75):IFB(A)=1THEN80
90 B(A)=1
100 IFA<=15THENP=4+32*A:PRINT@P,CHR$(255);:GOTO150
110 IFA<=30THENP=9+32*(A-15):PRINT@P,CHR$(255);:GOTO150
120 IFA<=45THENP=14+32*(A-30):PRINT@P,CHR$(255);:GOTO150
130 IFA<=60THENP=19+32*(A-45):PRINT@P,CHR$(255);:GOTO150
140 P=24+32*(A-60):PRINT@P,CHR$(255);
150 K$=INKEY$
160 KK=KK+1:IFKK=75THEN210
170 A$=INKEY$:IFA$=""THEN170
180 IFA$="Z"THENRUNELSEIFASC(A$)=13THEN190ELSE170
190 PRINT@P,CHR$(191);
200 GOTO80
210 CLS:PRINT@194,"YOU USED ALL THE SPACES!!!"
```

Program Listing 3

Program Listing 4

```
5 'REREPEAT
7 'JAMES W. WOOD, 424 N. MISSOURI, ATWOOD, IL, 61913
10 CLS
20 PRINT@230,"REPEAT!!!":INPUT"  HOW MANY CAN YOU REPEAT";M
30 IFM>50THENPRINT"SORRY, I CAN ONLY GIVE FIFTY":PRINT@281,"
  ";:GOTO20
40 IFM<1THENPRINT"DON'T BE SILLY";:PRINT@281,"       ";:GOTO20
45 PRINT@416,"   USE THE ARROWS TO REPEAT THE PATTERNS OF LIGHT
!!!!";
46 FORD=1TO1500:NEXTD
50 DIMQ(50):CLS0:FORB=1TO4:FORA=1TO5:A$(B)=A$(B)+CHR$(127+16*B):
NEXTA,B
60 FORW=1TO50:Q(W)=RND(4):NEXTW
70 E=E+1:IFE=M+1THENCLS0:FORI=1TO30:SOUNDRND(150)+50,1:PRINT@RND
(500),A$(RND(4));:NEXTI:GOTO250
80 FORF=1TOE:FORD=1TO100:NEXTD
90 CLS0:FORD=1TO100:NEXTD:ONQ(F)GOSUB210,220,230,240
100 NEXTF:FORD=1TO100:NEXTD:CLS0
110 FORF=1TOE:Z=0:CLS0
```

Listing 4 continues

See List of Advertisers on Page 563

Rerepeat is an imitation of Simon. Colored squares at the top, bottom, and sides of the screen flash a pattern that you must repeat using the up, down, left, and right arrows. First repeat one square, then the same one plus another, and so on. If you successfully repeat the pattern, a wild graphics display appears on the screen.

"Colored squares at the top, bottom, and sides of the screen flash a pattern..."

Hangman makes you guess the letters in a word in an attempt to save the little man from the rope. You can change the words in the data lines or add more. Be sure to change the 72 in line 70 to the total number of words in your data lines. ■

James Wood (424 N. Missouri, Atwood, IL 61913) is the coach for a high-school scholastic team.

Listing 4 continued

```
120 A$=INKEY$
130 A$=INKEY$:IFA$=""THEN130
140 IFA$=CHR$(94)ANDQ(F)=1THENGOSUB210:Z=1
150 IFA$=CHR$(8)ANDQ(F)=2THENGOSUB220:Z=1
160 IFA$=CHR$(9)ANDQ(F)=3THENGOSUB230:Z=1
170 IFA$=CHR$(10)ANDQ(F)=4THENGOSUB240:Z=1
180 IFZ=0THENFORV=1TO5:SOUND5,5:NEXTV:FORF=1TOE:ONQ(F)GOSUB210,2
20,230,240:FORD=1TO100:NEXTD:CLS0:NEXTF:GOTO250
190 NEXTF
200 GOTO70
210 FORC=13TO77STEP32:PRINT@C,A$(1);:NEXTC:SOUND147,2:RETURN
220 FORC=193TO257STEP32:PRINT@C,A$(2);:NEXTC:SOUND159,2:RETURN
230 FORC=218TO282STEP32:PRINT@C,A$(3);:NEXTC:SOUND170,2:RETURN
240 FORC=429TO493STEP32:PRINT@C,A$(4);:NEXTC:SOUND176,2:RETURN
250 CLS:PRINT@230,"PLAY AGAIN? (Y/N)"
260 B$=INKEY$:IFB$="Y"THENRUNELSEIFB$="N"THENENDELSE260
```

```
10 'HANGMAN
20 'JAMES W. WOOD,424 N. MISSOURI,ATWOOD, IL, 61913
30 CLS:DIMA$(30):FORA=1TO14:BL$=BL$+CHR$(128):NEXTA:FORA=201TO48
9STEP32:PRINT@A,BL$;:NEXTA
40 PRINT@235,CHR$(159)+CHR$(156)+CHR$(156)+CHR$(158)+CHR$(156)+C
HR$(156)+CHR$(157)+CHR$(156);
50 PRINT@267,CHR$(159)+CHR$(128)+CHR$(150);
60 PRINT@299,CHR$(159)+CHR$(150);:FORA=331TO491STEP32:PRINT@A,CH
R$(159);:NEXTA
70 T=RND(72):FORY=1TOT:READA$:NEXTY
80 B=LEN(A$):FORQ=1TOB:A$(Q)=MID$(A$,Q,1):NEXTQ
90 PRINT@96,"LETTERS USED,";:FORA=1TOB:PRINT@38+A,"-";:NEXTA
100 PRINT@160,"WHAT LETTER?";
110 PRINT@172," ";
120 IK$=INKEY$
130 B$=INKEY$:IFB$=""ORB$=CHR$(13)THEN130
140 PRINT@172,B$;:FORTI=1TO40:NEXTTI
150 FORQ=1TOB:IFB$=A$(Q)THENPRINT@Q+38,B$;:SOUND150,1
160 NEXTQ
170 FORQ=1TOB:IFB$<>A$(Q)THENNEXTQ:G=G+1:SOUND5,1:ON G GOSUB 210
,220,230,240,250,260,270,280,290,300
180 PRINT@W+110,B$;:W=W+1
190 FORQ=1TOB:IFCHR$(PEEK(1062+Q))=A$(Q)ORPEEK(1062+Q)=64+ASC(A$
(Q))THENNEXTQ:FORT=1TO10:PRINT@12,"CORRECT";:FORTT=1TO10:NEXTTT:
PRINT@12,"        ";:SOUND150,1:FORTT=1TO10:NEXTTT:NEXTT:RUN
200 GOTO100
210 PRINT@272,CHR$(158)+CHR$(159)+CHR$(157);:PRINT@304,CHR$(157)
+CHR$(159)+CHR$(158);:RETURN
220 PRINT@336,CHR$(143)+CHR$(143)+CHR$(143);:PRINT@368,CHR$(143)
+CHR$(143)+CHR$(143);:PRINT@400,CHR$(175)+CHR$(175)+CHR$(175);:P
RINT@432,CHR$(172)+CHR$(172)+CHR$(172);:RETURN
230 PRINT@432,CHR$(175);:PRINT@464,CHR$(175);:RETURN
240 PRINT@434,CHR$(175);:PRINT@466,CHR$(175);:RETURN
250 PRINT@335,CHR$(142);:PRINT@367,CHR$(138);:RETURN
260 PRINT@339,CHR$(141);:PRINT@371,CHR$(133);:RETURN
270 PRINT@399,CHR$(158);:RETURN
280 PRINT@403,CHR$(157);:RETURN
290 PRINT@495,CHR$(177)+CHR$(183);:RETURN
300 PRINT@498,CHR$(187)+CHR$(178);
310 PRINT@0," THE WORD IS ";CHR$(34);A$;CHR$(34):SOUND89,6:SET(3
3,17,2):SOUND89,8:RESET(33,17):SOUND89,4:SET(36,17,2):SOUND89,8:
RESET(36,17):SOUND117,8:SET(33,17,2):SOUND108,4:RESET(33,17):SOU
ND108,8:SET(36,17,2):SOUND89,4:RESET(36,17):SOUND89,8
320 SET(36,17,2):SOUND78,4:SET(33,17,2):SOUND89,8:RUN
330 DATAABLE,ABOUT,ACCIDENT,BABY,BALANCE,BED
340 DATACARE,CHANGE,CONTAINER,DAUGHTER,DELIVER,DISAPPOINTED
350 DATAEAT,EMPTY,ENERGY,FIGURE,FAST,FRUIT
360 DATAGATHER,GRACEFUL,GRANDFATHER,HANDKERCHIEF,HELLO,HUNGRY
370 DATAIODINE,ICE,INVENT,JACKET,JET,JUNGLE
380 DATAKANGAROO,KINGDOM,KINDERGARTEN
390 DATALADDER,LATER,LADY
400 DATAMONEY,MATCH,MARRY
410 DATANOTHING,NEW,NARROW
420 DATAOCTOPUS,OATMEAL,ONLY
430 DATAPACKAGE,PARTY,PALACE
440 DATAQUARTER,QUESTION,QUEEN
450 DATARACE,READY,REASON
460 DATASAD,SAILBOAT,SAT
470 DATATALL,TEACHER,TEASE
480 DATAUGLY,UNDERSTAND,UNUSUAL
490 DATAVACATION,VASE,VACUUM
500 DATAWADE,WAGON,WALL
510 DATAXYLOPHONE,ZEBRA,ZIPPER
```

Program Listing 5

Reconcile with Color

by David Dacus

You don't need a Model II, CP/M, and a general ledger program to balance your checkbook. Just run this routine on your CC with 32K.

If keeping track of finances gets you down, try this program. It contains a comprehensive financial management package associated with the check register, and it performs the following tasks:

- Enters transactions
- Edits transactions
- Reconciles your bank statement
- Searches for checks to a specific payee
- Searches for a series of checks
- Lists all checks in the file to a printer
- Shows the current bank balance
- Searches for checks written during a specific month
- Searches for checks written for a specific category
- Exits the program with no file save
- Records all checks to disk and exits

This program was written for a 32K Color Computer with Radio Shack Disk Basic, and it uses all the available memory in the 32K. With one page of graphics reserved when the program is running (PCLEAR 1—the smallest amount reservable with disk operational) and with all arrays dimensioned, 202 bytes are free. You can modify the program to operate on a 16K Color Computer or a TRS-80 Model I, II, or III. Suggestions for modifications and a discussion of some of the Color Computer's commands appear in the conclusion.

The program keeps track of all checks written on your account by asking you to enter five data items for each check. The program accepts credits and deposits as well as checks and debits. The current balance is maintained with each entry; this is useful if you have two checkbooks for the same account.

Another useful function is the search by category. This function is priceless at income-tax time.

Reconcile saves time when the monthly bank statement arrives. You can reconcile savings and checking accounts in 15 to 20 minutes; it takes two to three hours by hand.

Algorithms

The Enter subroutine (lines 260–410) lets you enter new checks, deposits, credits, and debits. The program asks you to enter a check by entering five data items separated by commas. Enter the data in the following order: check number, up to six digits; payee name, up to 16 characters; the amount of the check, up to nine characters; the month the check was written, three characters (e.g., JAN); and the category of the expenditure, up to 16 characters.

Any entry that adds money to the account must be entered as category Deposit. The automatic balance function searches the category of an entry for the value Deposit. If the category is Depos-

Program Listing

```
10 REM CHECK REGISTER
20 REM COPYRIGHT 1982 BY DAVID M. DACUS, 1670 VALENCIA, LAS CRUC
ES, NM 88001
25 GOTO 2005
30 CLEAR 7000
40 CLS:PRINT@0,STRING$(64,165):INPUT"IS THIS A NEW FILE (Y OR N)
";A$:PRINT@128,STRING$(32,165)
50 DIM FIL$(80),T(250),A(250)
60 DIM NAM$(250),CHK$(250),AMT$(250),MON$(251),CAT$(250),REC$(25
0)
70 IF A$="Y" THEN INPUT "WHAT IS THE NAME FOR YOUR        NEW DAT
A FILE?";FL$:N = 0:GOTO 200
80 INPUT "WHAT IS THE NAME OF YOUR        CURRENT DATA FILE?";FL
$:PRINT STRING$(32,165);:PRINT"LOADING DATA"
90 OPEN "D", #1, FL$, 64
100 N = N + 1
110 GET #1, N
120 INPUT #1, CHK$(N),AMT$(N),REC$(N),NAM$(N),MON$(N),CAT$(N)
130 A(N) = VAL(AMT$(N))
140 IF LOF(1) <> N THEN 100
150 CLOSE #1
160 FOR D = 1 TO N
170 IF CAT$(D) <> "DEPOSIT" THEN 190
180 T(D) = T(D-1) + A(D):NEXT D:GOTO 200
190 T(D) = T(D-1) - A(D):NEXT D
200 CLS:PRINT@0,STRING$(32,175)
210 PRINT@ 32,"**TO RECOVER FROM ERROR WITHOUT LOSING DATA ENTER
 GOTO 200**"
```

Listing continues

The Key Box

Color Computer
32K
Extended Color Basic
Color TRSDOS, 1 disk drive

Listing continued

```
220 PRINT @ 96, STRING$(32,175):PRINT @ 128, "SELECT A FUNCTION.
ENTER A NUMBER":PRINT @ 160, STRING$(32,175)
230 PRINT @ 192, "1.ENTER","2.EDIT","3.RECONCILE","4.PAYEE","5.C
HECK #","6.LIST","7.BALANCE","8.MONTH","9.CATEGORY","10.QUIT","1
1.EXIT - SAVE FILE":PRINT @ 384, STRING$(32,175)
240 INPUT "PLEASE ENTER A #";B:IF B < 1 OR B > 11 THEN FOR B = 1
 TO 200:PRINT "PLEASE ENTER A # BETWEEN 1 & 11":NEXT B:GOTO 200
250 ON B GOTO 260,420,600,990,1230,1360,1460,1490,1650,2000,1940
260 FOR D = N + 1 TO 250:CLS:PRINT@0,STRING$(32,169):PRINT@32, "
** ENTER ANY MONEY ADDED TO THE ACCOUNT AS CATEGORY - DEPOSIT **
":PRINT@96,STRING$(32,169)
270 PRINT@128,"LAST CHECK ENTERED WAS ";CHK$(N):PRINT@160,STRING
$(32,169)
280 PRINT@192, "ENTER CHECK DATA SEPARATED BY ,S (6)/(16)/ (9)
/ (3) /  (16)    CHK#,NAME,AMOUNT,MONTH,CATEGORY":PRINT@288,STRIN
G$(32,169)
290 INPUT CHK$(D),NAM$(D),AMT$(D),MON$(D),CAT$(D):A(D) = VAL(AMT
$(D)):IF N <= 1 THEN 350
300 FOR E = 1 TO N:IF CHK$(D) <> CHK$(E) THEN NEXT E:GOTO 350
310 CLS:PRINT@0,STRING$(224,246);"CHECK NUMBER ";CHK$(D);" HAS B
EEN      ENTERED ALREADY"
320 PRINT"CHECK NUMBER ";CHK$(D);" IS":PRINT CHK$(E)" "NAM$(E)"
"AMT$(E)" "MON$(E)" "CAT$(E)
330 INPUT "DO YOU WANT TO 1.RETURN TO INPUT 2.RETURN TO MAIN MEN
U"; B
340 IF B = 1 THEN 260 ELSE GOTO 200
350 N = D:IF CAT$(D) <> "DEPOSIT" THEN 370
360 T(D) = T(D-1) + A(D):GOTO 380
370 T(D) = T(D-1) - A(D)
380 CLS:PRINT@0,STRING$(32,169);"THERE ARE NOW";N;"CHECKS IN THE
  FILE.THE LIMIT IS 250 CHECKS":PRINT@96,STRING$(32,169)
390 PRINT"THE CURRENT BALANCE IS ":PRINTUSING"**$##,#####.##";T(
D):PRINT@224,STRING$(32,169)
400 INPUT "ARE YOU THROUGH ENTERING (Y/N)";X$:IF X$ <> "N" THEN
200
410 NEXT D:GOTO200
420 CLS:PRINT@0,STRING$(160,181);:INPUT "ENTER CHECK # FOR EDIT"
;X$:CLS:PRINT@0,STRING$(32,181);
430 PRINT @ 32, "CHK#,NAME,AMOUNT,MONTH,CATEGORY":PRINT@64,STRIN
G$(32,181);
440 FOR D = 1 TO N:IF X$ <> CHK$(D) THEN NEXT D
450 PRINT CHK$(D)" "NAM$(D)" "AMT$(D)" "MON$(D)" "CAT$(D):PRINT@
192,STRING$(32,181);
460 PRINT@224,"WHICH OF THE ITEMS IS INCORRECT ";"1.CHECK #","2.
NAME","3.AMOUNT","4.MONTH","5.CATEGORY","6.ALL ARE RIGHT":INPUT
B:CLS:PRINT@0,STRING$(32,181);
470 ON B GOTO 480,490,500,550,560,590
480 INPUT "ENTER CORRECT CHECK #";CHK$(D):GOTO570
490 INPUT "ENTER CORRECT NAME";NAM$(D):GOTO570
500 INPUT "ENTER CORRECT AMOUNT";AMT$(D):A(D) = VAL(AMT$(D))
510 FOR E = D TO N:IF CAT$(E) <> "DEPOSIT" THEN 530
520 T(E) = T(E-1) + A(E):NEXT E:GOTO 540
530 T(E) = T(E-1) - A(E):NEXT E
540 PRINT"THE CORRECT BALANCE IS":PRINT USING "**$##,####.##";T(
N):PRINT@128,STRING$(32,181);:GOTO 570
550 INPUT"ENTER CORRECT MONTH";MON$(D):GOTO 570
560 INPUT"ENTER CORRECT CATEGORY";CAT$(D)
570 PRINT @ 224,CHK$(D)" "NAM$(D)" "AMT$(D)" "MON$(D)" "CAT$(D):
PRINT@288,STRING$(32,181);
580 PRINT@320,;:INPUT "IS THE ENTRY CORRECT(Y/N)";X$:IF X$ = "N"
 THEN CLS:GOTO 450
590 PRINT@288,STRING$(32,181);:INPUT "DO YOU WANT TO EDIT AGAIN(
Y/N)";X$:IF X$ = "Y" THEN 420 ELSE 200
600 CLS:PRINT@0,STRING$(32,185);"ENTER# OF CHECK TO BE RECONCILE
D":INPUT "CHECK # ";X$
610 PRINT@96,STRING$(32,185);"CHK#,NAME,AMOUNT,MONTH,CATEGORY ";
STRING$(32,185)
620 FOR D = 1 TO N:IF X$ <> CHK$(D) THEN NEXT D
630 PRINT@192,CHK$(D)" "NAM$(D)" "AMT$(D)" "MON$(D)" "CAT$(D):PR
INT@256,STRING$(32,185)
640 PRINT"DOES THE COMPUTER ENTRY MATCH   YOUR CHECK ",,"1.YES",
"2.NO","3.DO NOT RECONCILE":INPUT "#";REC$(D)
650 IF VAL(REC$(D)) <> 2 THEN 790
660 CLS:PRINT@0,STRING$(32,185);CHK$(D)" "NAM$(D)" "AMT$(D)" "MO
N$(D)" "CAT$(D)
670 PRINT@96,STRING$(32,185);"WHICH OF THE ITEMS IS INCORRECT ";
"1.CHECK #","2.NAME","3.AMOUNT","4.MONTH","5.CATEGORY","6.ALL AR
E RIGHT":INPUT B:PRINT@288,STRING$(32,185);
680 ON B GOTO 690,700,710,750,760,790
690 INPUT "ENTER THE CORRECT CHECK #";CHK$(D):GOTO770
700 INPUT "ENTER THE CORRECT NAME";NAM$(D):GOTO770
710 INPUT "ENTER THE CORRECT AMOUNT";AMT$(D):A(D) = VAL(AMT$(D))
720 FOR E = D TO N:IF CAT$(E) <> "DEPOSIT" THEN 740
730 T(E) = T(E-1) + A(E):NEXT E:GOTO770
740 T(E) = T(E-1) - A(E):NEXT E:GOTO 770
```

Listing continues

it, the amount is added to the account. All other entries are subtracted from the account.

Enter's algorithm prevents you from entering two checks by the same check number. If two checks are entered by the same number, you can never reach the second check to reconcile it. If you need to enter two checks with the same number, label one with a trailing alpha character (e.g., 1459 and 1459A).

Each time you enter a check you are asked if you want to enter another check or return to the main menu. The default value is return to the main menu, and if you press enter without entering a response, the main menu appears.

Edit (lines 420–590) allows you to change any data item entered for a check. When you enter a check number, the current data for that check displays and you are offered six choices. Options 1–5 let you change one of the five data entries. Option 6 lets you say that the data is fine. This option is offered in case you select the wrong check number for edit.

If you select option 6, you do not need to change any value. You are asked if you want to edit another check or return to the main menu. If you change a data entry, the new check displays and you are asked if it is correct. If you answer yes, the program asks if you wish to enter another check for edit. If the check is not correct, it asks which data item you wish to change. When the data item changed is the amount, the balance of the account is updated.

Reconcile (lines 600–980) is the heart of the checkbook portion of the program. This algorithm keeps track of each check that has been returned with your monthly bank statement, so you know which checks have cleared the bank and which have not.

The function asks for a check number to be reconciled. When you enter the number, the data stored for the entered check is displayed to the screen and you are asked if the data matches the check you hold in your hand. If you enter 1 (yes) the check is flagged as reconciled and will not be listed as an outstanding check. If you enter 2 (no) you are sent to an edit subroutine and asked which data item is incorrect.

This subroutine works in the same manner as the Edit function. After you have corrected the data you are again asked if the data matches the check. If you respond with a 1 or 2, the procedures just discussed are repeated. Option 3 is "do not reconcile." This

option allows you to skip functions 1 and 2 if you have entered an erroneous check number or do not wish to reconcile the check.

After you have completed option 1, 2, or 3, you are asked if you want to enter another check, list all outstanding checks, or return to the main menu. If you request to enter another check, repeat the above procedure. If you ask to return to the main menu, you exit the reconcile algorithm.

If you ask to list all outstanding checks, you are offered two options. You can have all outstanding checks listed on the printer or on the screen. Once you have selected either a printout or CRT listing, you are asked for the closing balance of the current bank statement. All five data items are listed for each outstanding check, followed by a list of the total value for the outstanding checks and a list of the value of outstanding deposits. The reconciled balance is also listed.

The Payee function (lines 990–1220) lists all checks to a specified payee. You are first presented with the option of listing all payees. The payees may either be listed to the CRT or to the printer. If you have written checks to a large number of payees, you may experience a BS error. The unique-payees array is dimensioned to hold 80 payees (FIL$(80) in line 50). After the payees have been listed, or if you select no listing, you are asked to name the payee you want listed. You can then select between listing all checks to the payee to either the printer or the CRT.

The Check # function (lines 1230-1350) lets you list a series of checks searched by check number. You are asked to input the first and last number for the search, separated by a comma.

List (lines 1360–1450) sends a list of all checks in the file to the printer. At the end of the check list, the current bank balance is printed.

The Balance function (lines 1460–1480) lists the current balance to the screen.

Month (lines 1490–1640) lists all checks written within a single month.

Category (lines 1650–1930) automatically provides a list of categories and then asks you to select a category for Search.

The Quit function (line 2000) exits the program without saving any transactions done before the exit.

The Exit-Save File function (lines 1940–2000) saves all check data to the disk file named when you enter the program, and then exits the program. You

Listing continued

```
750 INPUT "ENTER THE CORRECT MONTH";MON$(D):GOTO770
760 INPUT "ENTER THE CORRECT CATEGORY";CAT$(D)
770 CLS:PRINT@0,STRING$(32,185);"THE CURRENT BALANCE IS":PRINT U
SING"**$##,#####.##";T(N):PRINT@96,STRING$(32,185)
780 PRINT@128,CHK$(D)" "NAM$(D)" "AMT$(D)" "MON$(D)" "CAT$(D):PR
INT@192,STRING$(32,185);:INPUT "IS THE CHECK DATA CORRECT";X$:IF
 X$ = "N" THEN 660 ELSE GOTO640
790 PRINT@256,STRING$(32,185);:PRINT "DO YOU WANT TO ",,"1.ENTER
 ANOTHER CHECK","2.LIST ALL CHECKS OUTSTANDING","3.ENTER ANOTHER
 FUNCTION";STRING$(41,32);:INPUT E
800 ON E GOTO 600,810,200
810 CLS:PRINT@224,STRING$(32,185):INPUT"DO YOU WANT (1)HARDCOPY,
 OR      (2)CRT LISTING OF OUTSTANDING   CHECKS?";F:INPUT"ENTER C
LOSING BALANCE FROM       STATEMENT";S1:K=0:H=0:IF F = 2 THEN 910
820 PRINT "LISTING OUTSTANDING CHECKS":PRINT#-2,"****LIST OF OUT
STANDING CHECKS*";CHR$(10);CHR$(10)
830 PRINT#-2,"CHECK #","MONTH","NAME","AMOUNT","CATEGORY"
840 FOR D = 1 TO N
850 IF VAL(REC$(D)) <> 1 THEN PRINT #-2, CHK$(D),MON$(D),NAM$(D)
,AMT$(D),CAT$(D):IF CAT$(D) = "DEPOSIT" THEN 860 ELSE H = H + A(
D)
860 IF VAL(REC$(D)) <> 1 AND CAT$(D) = "DEPOSIT" THEN K = K + A(
D)
870 NEXT D:PRINT #-2, CHR$(10);CHR$(10);"THE TOTAL VALUE OF OUTS
TANDING CHECKS IS $"H
880 IF K > 0 THEN PRINT #-2, CHR$(10);"THE TOTAL VALUE OF OUTSTA
NDING DEPOSITS IS $"K
890 S1=S1+K:S1=S1-H:PRINT#-2,CHR$(10);CHR$(10);"* RECONCILED BAL
ANCE= ";S1
900 GOTO200
910 PRINT"CHK#,NAME,AMOUNT,CATEGORY":FOR D = 1 TO N
920 IF VAL(REC$(D)) <> 1 THEN PRINT CHK$(D)" "NAM$(D)" "AMT$(D)"
 "CAT$(D) ELSE 940
930 IF CAT$(D) <> "DEPOSIT" THEN H = H + A(D) ELSE K = K + A(D)
940 NEXT D
950 PRINT"THE TOTAL VALUE OF OUTSTANDING  CHECKS IS:   "H
960 IF K > 0 THEN PRINT "THE VALUE OF OUTSTANDING        DEPOSIT
S IS " K
970 S1=S1+K:S1=S1-H:PRINT"RECONCILED BALANCE="S1
980 INPUT "PRESS ENTER TO RETURN";X$:GOTO 200
990 CLS:PRINT@0,STRING$(32,201):INPUT "DO YOU NEED A LIST OF PAY
EES";X$:IF X$ = "Y" THEN INPUT "HARDCOPY (Y/N)";X1$:GOSUB 1120:I
F X1$ <> "Y" THEN INPUT "ARE YOU FINISHED READING";X$:CLS ELSE C
LS
1000 PRINT@0,STRING$(32,201);:INPUT "ENTER PAYEE TO BE SOUGHT
   ";X$:PRINT@128,STRING$(32,201);:INPUT "DO YOU WANT HARD COPY(
ENTER 1)";E:IF E <> 1 THEN 1080
1010 CLS:PRINT@0,STRING$(160,201):PRINT"PRINTING PAYEE LISTING"
1020 PRINT#-2, "PAYEE - "X$,CHR$(10),CHR$(10)
1030 PRINT#-2, "CHECK #","MONTH","AMOUNT","CATEGORY",CHR$(10):X1
 = 0:FOR D = 1 TO N
1040 IF X$ <> NAM$(D) THEN 1060
1050 PRINT#-2, CHK$(D),MON$(D),AMT$(D),CAT$(D):X1 = X1 + VAL(AMT
$(D))
1060 NEXT D
1070 PRINT#-2, CHR$(10),CHR$(10);"THE TOTAL PAID TO ";X$;" IS ";
X1:GOTO 200
1080 CLS:PRINT "CHECK#,MONTH,AMOUNT,CATEGORY"
1090 X1 = 0:FOR D = 1 TO N:IF X$ <> NAM$(D) THEN 1110
1100 PRINT CHK$(D)" "MON$(D)" "AMT$(D)" "CAT$(D):X1 = X1 + VAL(A
MT$(D))
1110 NEXT D:PRINT"** THE TOTAL PAID TO ",X$;" IS";X1:INPUT "ARE
YOU THROUGH READING";X$:GOTO200
1120 E = 1:X1 = 0:FIL$ = ""
1130 FOR D = 1 TO N
1140 FOR X = 1 TO E
1150 IF FIL$(X) = NAM$(D) THEN 1210
1160 NEXT X
1170 IF X1$ = "Y" THEN PRINT#-2, NAM$(D),:GOTO 1190
1180 PRINT NAM$(D),
1190 FIL$(E) = NAM$(D)
1200 E = E + 1
1210 NEXT D
1220 RETURN
1230 CLS:PRINT@ 128, STRING$(32,246):INPUT "PLEASE ENTER FIRST A
ND LAST      CHECK NUMBER FOR SEARCH (F,L)   ";X$,X1$
1240 CLS:PRINT@ 128,STRING$(32,246):INPUT "DO YOU WANT HARDCOPY(
Y/N)";X2$:IF X2$ <>"Y" THEN 1310
1250 CLS:PRINT@ 128,STRING$(32,246);"PRINTING CHECK SERIES"
1260 PRINT#-2, "LISTING OF ALL CHECKS NUMBERED  BETWEEN ";X$;" A
ND ";X1$;CHR$(10);CHR$(10)
1270 PRINT#-2, "CHECK #","PAYEE","AMOUNT","MONTH","CATEGORY";CHR
$(10)
1280 FOR D = 1 TO N:IF VAL(CHK$(D)) < VAL(X$) OR VAL(CHK$(D)) >
VAL(X1$) THEN NEXT D:GOTO200
```

Listing continues

are also asked if you intend to exit. If you do not answer Y you are returned to the main menu.

I use this option to back up my check files. I keep two disks current with my checks. When I exit saving the file to disk, I tell the computer I do not want to exit. Then, I switch disks and re-exit, saving my file, and copying data to my back-up disk.

These functions should meet most of the needs for financial transactions for a family or small business, but the program is easy to tailor to your own needs.

To adapt the program for a 16K machine, reduce the number of checks stored per file to 50 or 75. You may also want to eliminate some of the functions, such as Edit (Reconcile has an edit function anyway), Payee, Check #, and Month.

Those of you who don't have a Color Computer will have to reformat the displays, because the Color Computer has a display of 14 lines of 32 characters rather than 16 of 64 or 24 of 80. All the PRINT@ commands, except those of PRINT@0, must be recalculated. PRINT STRING$(X,Y) prints a string of X characters of ASCII code Y. The Y values in this program are block color graphics, which can be replaced on the Models I, II, and III by the block-graphic characters for these machines.

The disk commands in lines 90 and 1940 open a direct (random access) file with a limit of 64 characters per entry. The other computers require appropriate random-access file commands and format statements. The Color Computer has no DEFINT command. Some arrays can be converted to integers to save memory. The PRINT #−2 command of Color Basic is the equivalent of LPRINT. The Print Using format "**$###,###.##" specifies two decimal places with commas inserted every three numbers, a floating dollar sign, and fill from the left with asterisks. Color Basic doesn't allow the defining of a format variable. If your computer lets you define a format, you don't have to repeat the format each place it is used.

The program is long, but it should make your financial management tasks easier. The program also makes use of some unusual features of the Color Computer that you may find useful in your programming efforts. ■

David Dacus (1670 Valencia, Las Cruses, NM 88001) is an Operations Research Analyst for the U.S. Army Training and Doctrine Command Systems Analysis Activity at White Sands Missile Range, NM.

Listing continued

```
1290 PRINT#-2, CHK$(D),NAM$(D),AMT$(D),MON$(D),CAT$(D)
1300 NEXT D:GOTO 200
1310 CLS:PRINT"LIST OF CHECKS ";X$" TO ";X1$
1320 PRINT "CHK#,NAME,AMOUNT,MONTH,CATEGORY"
1330 FOR D = 1 TO N:IF VAL(CHK$(D)) < VAL(X$) OR VAL(CHK$(D)) >
VAL(X1$) THEN NEXT D:GOTO 1350
1340 PRINT CHK$(D)" "NAM$(D)" "AMT$(D)" "MON$(D)" "CAT$(D):NEXT
D
1350 INPUT "ARE YOU THROUGH WITH THE LIST";X$:GOTO 200
1360 CLS:PRINT@0,STRING$(192,134);"PRINTING A LIST OF ALL CHECKS
   IN THE FILE":PRINTSTRING$(192,134)
1370 PRINT#-2,"LIST OF ALL CHECKS"
1380 PRINT#-2,CHR$(10)
1390 PRINT #-2, "CHECK #","MONTH","NAME","AMOUNT","CATEGORY"
1400 FOR D = 1 TO N
1410 PRINT #-2, CHK$(D),MON$(D),NAM$(D),AMT$(D),CAT$(D)
1420 NEXT D
1430 FOR D = 1 TO 3:PRINT #-2,CHR$(10):NEXT D
1440 PRINT#-2,"CURRENT BALANCE IS:  ";:PRINT#-2, USING "**$##,##
##.##";T(N)
1450 GOTO 200
1460 CLS:PRINT@0,STRING$(192,172);
1470 PRINT"CURRENT BALANCE IS:";:PRINTUSING "**$##,####.##";T(N)
:PRINT@256,STRING$(160,172);
1480 INPUT"READY TO RETURN";X$:GOTO200
1490 CLS:PRINT@0,STRING$(32,206);:INPUT"ENTER MONTH FOR FILE SEA
RCH";A$:PRINT@96,STRING$(32,206);
1500 INPUT"DO YOU WANT HARDCOPY(ENTER 1)";E:IF E<>1 THEN1580
1510 PRINT@192,STRING$(32,206);"PRINTING CHECKS FOR "A$:PRINT#-2
, "******** ";A$;" ********";CHR$(10);CHR$(10)
1520 PRINT#-2,"CHECK #","NAME","AMOUNT","CATEGORY"
1530 PRINT#-2,CHR$(10)
1540 FOR D = 1 TO N
1550 IF A$ <> MON$(D) THEN 1570
1560 PRINT #-2, CHK$(D),NAM$(D),AMT$(D),CAT$(D)
1570 NEXT D:GOTO 200
1580 CLS:PRINT@0,STRING$(32,206);
1590 PRINT"CHECK #, NAME, AMOUNT, CATEGORY"
1600 FOR D = 1 TO N
1610 IF A$ <> MON$(D) THEN NEXT D:GOTO 1640
1620 PRINT CHK$(D);" ";NAM$(D);" ";AMT$(D);" ";CAT$(D)
1630 NEXT D
1640 INPUT "READY TO RETURN";A$:GOTO 200
1650 GOSUB 1840
1660 CLS:PRINT@0,STRING$(32,239);:INPUT"ENTER CATEGORY FOR FILE
SEARCH ?";Z$
1670 PRINT@96,STRING$(32,239);
1680 INPUT"DO YOU WANT HARDCOPY(ENTER 1)";E:IF E<>1 THEN 1780
1690 PRINT@192,STRING$(32,239);"PRINTING CHECKS IN CATEGORY
 ";Z$:PRINT#-2,"CATEGORY - "Z$
1700 PRINT #-2, CHR$(10);"CHECK #","MONTH","NAME","AMOUNT";CHR$(
10)
1710 FOR D = 1 TO N
1720 IF Z$ <> CAT$(D) THEN 1750
1730 PRINT #-2,CHK$(D),MON$(D),NAM$(D),AMT$(D)
1740 X1 = X1 + VAL(AMT$(D))
1750 NEXT D
1760 PRINT#-2, CHR$(10);CHR$(10);CHR$(10);"THE TOTAL EXPENDITURE
S FOR CATEGORY ";Z$;" IS " X1
1770 GOTO 200
1780 CLS:PRINT@0,STRING$(32,239);:PRINT"CHECK #, MONTH, NAME, AM
OUNT"
1790 FOR D = 1 TO N
1800 IFZ$<>CAT$(D) THEN 1820
1810 PRINT CHK$(D)" "MON$(D)" "NAM$(D)" "AMT$(D)
1820 NEXT D
1830 INPUT "READY TO RETURN";Z$:GOTO 200
1840 CLS:E=1:X1 = 0:FIL$ = ""
1850 FOR D = 1 TO N
1860 FOR X = 1 TO E
1870 IF FIL$(X) = CAT$(D) THEN 1920
1880 NEXT X
1890 PRINT CAT$(D),
1900 FIL$(E) = CAT$(D)
1910 E = E + 1
1920 NEXT D
1930 INPUT "ARE YOU THROUGH READING";X1$:RETURN
1940 OPEN "D", #1, FL$, 64
1950 FOR I = 1 TO N
1960 WRITE #1, CHK$(I),AMT$(I),REC$(I),NAM$(I),MON$(I),CAT$(I)
1970 PUT #1, I
1980 NEXT I
1990 CLOSE #1
2000 CLS:PRINT@0,STRING$(224,246);:INPUT"ARE YOU SURE YOU WANT T
O EXIT";X$:IF X$ <> "Y" THEN 200 ELSE END
2005 PCLEAR 1:GOTO 30
```

Expanded Color Capabilities

by James Wood

According to the *Going Ahead With Extended Color Basic* manual, the Color Computer can display only certain colors on the screen at one time. In the high-resolution modes you can have black and green; black and buff; red, blue, green, and yellow; or cyan, magenta, orange, and buff. But, I wanted several colors on a black background. After experimenting, I obtained these colors. I don't know why it works, but I can tell you how to put green, red, blue, and black in the same picture (great colors for a space game).

The Color Computer can now display green, blue, red, and black on the screen at one time.

Type in and run the Red, Green, Blue, Black demo, which starts at line 295. Lines 300–330 use PMODE 3,1 and SCREEN 1,0 to create red, blue, and yellow boxes on a green background. After the For loop in line 340, the monitor is changed by PMODE 4,1:SCREEN1,1. The red changes to green, blue to red, yellow to blue, and green to black.

You could use these colors to create a space ship as well as some rocks to blow up. In the Space Rocks program, a red and blue ship shoots green laser blasts at red and blue rocks against a black sky. The rocks explode into green, red, and blue dust. The right joystick controls the up, down, left, and right movement of the space ship. Depressing the fire button creates a laser blast.

The program has one minor problem. The ship points to the right and shoots to the right. If a rock is on your left, you can still blow it up by shooting to the right. This problem will give you something with which to experiment. ■

James Wood can be reached at 424 N. Missouri, Atwood, IL 61913.

8	creates strings for numbers to display number of hits
10	clears screen, prevents FC error first time rock is erased
20	resets timer
30	dimensions array for ship and dust
50	draws ship
60	creates dust for explosion
70	paints ship
80	part of ship
90	changes colors
100	places ship into array
110	places dust into array
120	clears graphics and shows new graphics screen
130	reads joysticks
140	used with line 240
150	changes joystick reading to even numbers to prevent distorted ship image (necessary due to switching PMODEs)
160	places ship array onto screen
170	reads joysticks
180	displays laser blast and sound if joystick button is pressed
190	determines if rock has been hit, sounds explosion
200	delay to leave rock on screen long enough to hit (if you're fast)
210	erases rock
220	determines position for next rock
230	puts rock on screen
240	erases old ship position only if joysticks have moved (prevents flicker of motionless ship)
260	determines if number of hits equals 10, places number of hits on screen, prints time elapsed after 10 ships destroyed

Table 1. Line Descriptions

The Key Box

Color Computer
16K RAM
Extended Color Basic
One Joystick

Program Listing

```
5 'SPACE ROCKS
7 'JAMES W. WOOD, 424 N. MISSOURI, ATWOOD, IL, 61913
8 A$(1)="D1Ø":A$(2)="R5D5L5D5R5":A$(3)="R5D5L5R5D5L5":A$(4)="D5R
5U5D1Ø":A$(5)="R5L5D5R5D5L5":A$(6)="R5L5D1ØR5U5L5":A$(7)="R5D1Ø"
:A$(8)="R5D1ØL5U1ØD5R5":A$(9)="R5D5L5U5R5D1Ø"
1Ø U=3:V=3:CLSØ
2Ø TIMER=Ø
```

Listing continued

Listing continued

```
30 DIMW(2Ø,1Ø),E(1Ø,1Ø)
4Ø PMODE3,1:PCLS
5Ø DRAW"BMØ,ØR4F4R8F4L2ØU8"
6Ø FORT=1TO2Ø:PSET(RND(1Ø)+1ØØ,RND(1Ø)+1ØØ,RND(3)+1):NEXTT
7Ø PAINT(3,4),3,4
8Ø FORX=1ØTO14:PSET(X,6,2):NEXTX
9Ø PMODE4,1
1ØØ GET(Ø,Ø)-(2Ø,1Ø),W
11Ø GET(1ØØ,1ØØ)-(11Ø,11Ø),E
12Ø PCLS:SCREEN1,1
13Ø A=JOYSTK(Ø):B=JOYSTK(1)
14Ø A1=A:B1=B
15Ø A=INT(A/2)*2:B=INT(B/2)*2
16Ø PUT(A*4,B*3)-(A*4+2Ø,B*3+1Ø),W
17Ø AA=JOYSTK(Ø):BB=JOYSTK(1)
18Ø P=PEEK(6528Ø):IFP=126ORP=254THENPLAY"O4V31L255BCB":LINE(A*4+
22,B*3+8)-(255,B*3+8),PSET:LINE(A*4+22,B*3+8)-(255,B*3+8),PRESET
:GOTO19ØELSE2ØØ
19Ø IFABS((B*3+8)-V)<=3THENPUT(U-5,V-5)-(U+5,V+5),E:PLAY"O2L1ØØV
31GECA":LINE(U-1Ø,V-1Ø)-(U+1Ø,V+1Ø),PRESET,BF:NH=NH+1:GOSUB26Ø:G
OTO22Ø
2ØØ JW=JW+1:IFJW=6THENJW=Ø:GOTO21ØELSE24Ø
21Ø LINE(U-3,V-3)-(U+3,V+3),PRESET,BF
22Ø U=RND(1ØØ)+15Ø:V=RND(17Ø)+1Ø
23Ø CIRCLE(U,V),3
24Ø IF AA<>A1 OR BB<>B1 THENLINE(A*4,B*3)-(A*4+22,B*3+1Ø),PRESET
,BF:GOTO13Ø
25Ø GOTO17Ø
26Ø IFNH=1ØTHENPRINT@192,INT(TIMER/6Ø);"SECONDS FOR 1Ø HITS"ELSE
LINE(Ø,18Ø)-(6,19Ø),PRESET,BF:DRAW"BM1,18Ø"+A$(NH):RETURN
29Ø END
295 'RED,GREEN,BLUE,BLACK DEMO
298 'JAMES W. WOOD, 424 N. MISSOURI, ATWOOD, IL, 61913
3ØØ PMODE3,1:PCLS:SCREEN1,Ø
31Ø LINE(2Ø,2Ø)-(8Ø,8Ø),PSET,BF
32Ø COLOR3,1:LINE(8Ø,8Ø)-(14Ø,14Ø),PSET,BF
33Ø COLOR2,1:LINE(14Ø,14Ø)-(19Ø,19Ø),PSET,BF
34Ø FORTT=1TO5ØØ:NEXTTT
35Ø PMODE4,1:SCREEN1,1
36Ø GOTO36Ø
```

80 Index, 1980–1982

Many of you have asked for it, and here it is: a cumulative, annotated, cross-referenced *80 Micro* article index through October 1982. (Look for the rest of 1982 in the January issue). We've even included references for Debugs!

The index is divided into the following categories: Business, Education, Games, General, Graphics, Hardware, Home/Hobby, Programming Techniques, Science/Math, Tutorials, and Utilities. Reviews are listed in a section of their own. The articles are sorted by title.

The computer the article was written for is listed in parentheses at the end of the annotation. If no computer appears, it is a general article that applies to all TRS-80s.

You will notice that most are listed as Model I articles. Many of these programs will run on a Model III, but we could not test them all to be sure. Also, it is often easy to convert Basic programs from one TRS-80 to another.

If you do not have the issue that has the article you want, many back issues are still available. Look for an order form elsewhere in this magazine. ■

1980

BUSINESS

EDUCATION

GAMES

Title	Author	Issue/Page	Debug	Annotation
4K Intelligence	Lopez	3/80:55		Artificial intelligence in Level I. (Model I)
Asteroid Adventure	Perry/Taylor	10/80:212		4K arcade space game. (Model I)
Ball Box	Lewis	4/80:60		Based on Pong Squot. (Model I)
Code Cracker	Morgan	8/80:110		Generate codes for one or two players to decipher. (Model I)
Computer Monopoly	Adams	11/80:83		Computer simulation of the board game. (Model I)
Game of Life, The	Kitsz	6/80:38	8/80:16	Based on British mathematician John Conrey's game, in Assembly language. (Model I)
Heartbeat Away, A	Morey	8/80:76		Simulation for being President of U.S. (Model I)
Life in the Fast Lane	Kepner/Grace	8/80:62		Basic/Assembly game of Life. (Model I)
Ping-Pong	Moehlis	9/80:216		Hit the bouncing dot. (Model I)
Puzzler	Morgan	10/80:198		Find the hidden word. (Model I)
Rock, Scissors, Paper	Harris	4/80:116		An old standard computerized in 4K. (Model I)
Slot Machine	Fason	8/80:84		Casino simulation. (Model I)
Starfighter	Ferrera	8/80:59		Space arcade game for 4K. (Model I)
Swords and Sorcery II	Adams	8/80:42	10/80:14	Adventure game. (Model I)
Third Dimension, The	Dillehay	7/80:152	4/81:22	Tic-Tac-Toe-like game. (Model I)
True or False?	Krutch	6/80:116		Guessing game with artificial-intelligence techniques employed. (Model I)
U-Boat	Borrmann	8/80:68		Sink subs and learn matrix handling. (Model I)
Westward Ho!	Herold	10/80:148		Simulation of a westward trek by pioneers. (Model I)

GENERAL

Title	Author	Issue/Page	Debug	Annotation
Computer Bulletin Boards	Cambron	5/80:110		All about bulletin boards and how to access them.
Confessions of a Computer Derelict	Kornfeld	4/80:30		Humor
Courts Smashed Copyright?	Kitsz	10/82:55		Software Copyright and the courts.
Dealer's Experience, A	DeFonzo	4/80/98		RS dealer experience.
EDTASM Index	Kepner	6/80:124		A cross-referenced index for the RS E/A manual.
Electronic Networks	Robertson	11/80:62		Computers and networking.
How the Gamesman Began	Robertson	7/80:48		Scott Adams profile.
I Ching	Scarpelli	4/80:123		Calculate life's changes on the TRS-80. (Model I)
In the Beginning	Herro	7/80:78		One man's experience with the TRS-80. (Model I)
On the Radio	Hastings	8/80:86		Do a broadcast log on the TRS-80. (Model I)
Owner's Tale, An	Dilbeck	5/80:107		Why I bought my microcomputer.
Part-time Consultant	Morin	3/80:79		How to be a computer consultant.
Put-N	O'Brien	5/80:46		A list of structured statements to build a language around.
Radio Shack vs. Competition	Busch	11/80:109		TRS-80 compared to other computers.
Table of Contents, The	Thurlow	11/80:234		Outline for Scripsit instruction tapes.
Tally with an 80	Graham/Haller	11/80:114		Computerize election returns. (Model I)
Tandy Story, The	Brown	1/80:28		Short history of the Tandy/Radio Shack Corporation.
The Bottom Shelf	Shuford	2/80:36		Profile of a software company.

GRAPHICS

Title	Author	Issue/Page	Debug	Annotation
Adventures in Roseland	Joffe	6/80:62	8/80:16	Use J = a SIN X to draw a three-leafed rose. (Model I)
Basic Drawing	Gorsky	7/80:140		Draw on the screen and save your work. (Model I)
Beginner's Formatting	Keller	7/80:108		See Tutorials.
Beyond Blackjack	Thorsen	1/80:60		See Programming Techniques.
CAL81	Strazzarino	12/80:128		See Home/Hobby
Compu-Sketch	Hendricks	12/80:255		Short Basic program to draw on the screen. (Model I)
Curve Plotter	Cecil	7/80:130		Draw curved graphics in Basic. (Model I)
Doodle Bug	Bishop	6/80:72	8/80:16	Doodle on your screen with eight direction keys. (Model I)

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Double-size Graphics	Thiel	6/80:106		How to get around CHR$(23) and keep your graphics. (Model I)
Graphics Coder, The	Racine	8/80:160		See Utilities.
Images	Gorsky	11/80:220		Draw radial lines on the screen. (Model I)
Inside-out Debugging	Ogren	11/80:222		See Tutorials.
Kaleidopen	Nicholas	6/80:78		Display kaleidoscope-like graphics on the screen. (Model I)
Random Walker	Strazzarino	11/80:179		Randomly generated graphics on the screen. (Model I)
Real-time Graphics	Zidonis	6/80:82		Use PEEK and POKE in game simulations. (Model I)
Seasons Greetings	Vann	12/80:112	3/81:14	Generate greeting cards on your screen. (Model I)
Simple Graphics	Hommel	7/80:76		Draw designs with arrow keys in 4K. (Model I)
Super Graphics	Moyer	10/80:202	1/81:28	Fast animation using the Print graphics method. (Model I)

HARDWARE

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
300-baud Terminal	Loos	8/80:136		Use unused ICs in the TRS-80 to make it a half-duplex terminal. (Model I)
Babybug Keypad	Kitsz	3/80:88		Hex keypad to use with machine code. (Model I)
Box It In	Zalnerunas	3/80:132		Build a box to hold wires and keyboard. (Model I)
Build a Light Pen	Holder	4/80:38	6/80:12	(Model I)
Build Your Own Port	Hawkes/Reese	9/80:116		Build a parallel port using the 8255 PPI chip. (Model I)
Cassette Problems	Stoner/Barker	1/80:62		See Tutorials.
Cassette Problems II	Stoner/Barker	2/80:54		Build the Data Dubber, which regenerates original CSAVE pulses. (Model I)
Caveat Emptor	Parris	10/80:122		Trials and errors of homebrew interfacing. (Model I)
Cheap CLOAD Fix	King	1/80:132	4/80:27	Device to block unwanted noise on CLOADs. (Model I)
Cheap Video	Fowler/Murray	8/80:116		Interface a tv to your computer. (Model I)
CLOAD Micrometer	Thiel	5/80:84		Device to monitor load level.
CTR-41 Modifications	Hinrichs	4/80:110		Mods for a better recorder. (Model I)
Destick Your Relay	Lukoff	5/80:74		Mod to prevent relay from sticking on the cassette recorder.
Disaster Saver	Brooks	7/80:156		Power interlock to prevent garbaged disk files. (Model I)
DVM Interface for the 80	Casper/Freedman	10/80:156		Hook up a digital voltmeter to the TRS-80. (Model I)
Faster, Faster	Kitsz	2/80:50		Program speedup device. (Model I)
Fuse Fix	Winter	6/80:154		Put an external fuse on your power supply. (Model I)
H14, Meet the TRS-80	Friesen	10/80:118		Interface Heathkit printer to TRS-80. (Model I)
Homebrew Interface, A	Vince	3/80:96		Real-time interface for home climate control, etc. (Model I)
Homebrew Memory	Ragucci	5/80:70		Extra memory with no expansion interface. (Model I)
Homebrew TRS-80	Steele	11/80:146		Build your own TRS-80 from scratch. (Model I)
I/O Ports Plus	Harron	3/80:120		Interface to turn on any memory location for control or monitor use. (Model I)
Interfacing the NEC Spinwriter	Kunzman	10/80:144		Get the NEC printer to work at 1,200 baud. (Model I)
Joystick City	Suter	12/80:186		Interface joysticks via the cassette I/O port. (Model I)
Level II to Model 33	Colby	1/80:70	3/81:14	Interface Model 33 Teletype to TRS-80.
Look, a Super Snubber	Martel	11/80:216		Protect cassette's relay from high voltage across its contact when turned off.
Lowercase and Uppercase	Stoner/Barker	3/80:72		Build a lowercase mod. (Model I)
LPRINT "Cheap"	Blechman	2/80:100		Using a serial printer with TRS232/Formatter. (Model I)
Mem Size. . .20K?	Stanley	11/80:116	5/81:22	Add 4K to 16K. (Model I)
Mork & Mindy Monitor	Jackson	4/80:58		Use tv as a computer monitor. (Model I)
PR-40 Printer Interface	Hise	8/80:152		Connect an SWTPC printer to a TRS-80. (Model I)
Regulate It!	Klungle	3/80:113		Build a voltage regulator.
Relay Assistant	Jahns	7/80:112		Circuit to prevent recorder on/off control from welding in the close position.

Title	Author	Issue/Page	Debug	Annotation
Relay Protection	Richardson	1/80:78		Build a protection device for your cassette relay. (Model I)
Reverse Video	Kitsz	4/80:54	5/80:28	Build a reverse-video device. (Model I)
Selectric Hardcopy	Bickerton	9/80:102	11/80:22	Interface a TRS-80 and a Selectric typewriter. (Model I)
Serial Clank on the Print	O'Brien	10/80:194	4/81:22	Interfacing serial printers via the RS-232. (Model I)
Simple Interface	Mullin	2/80:94		Interface switches using one chip and eight resistors. (Model I)
Teletype Interface	Noeth	6/80:142		Interface a Model 33 with no modification to the CPU. (Model I)
Teletype Interface	Commander	9/80:84		Instructions and driver program to interface a TTY to your computer. (Model I)
Testing 1, 2, 3	Nelson	6/80:136		Interface your computer with a test measuring system. (Model I)
TTY Interface	Rumbolt	7/80:144		Use a TTY as a printer. (Model I)
Turn-on	Nestor	12/80:208		Automatically turn on the RS line printer with a relay and software control. (Model I)
Two Basics Better than One	Erickson	10/80:182		Mod for Level I and Level II in the same machine. (Model I)
Two Different Worlds	Eckert	7/80:116		A/D converter using the AD570 and AD582 chips. (Model I)
Video Tune-up	Miller	2/80:62		Eliminate fuzzy edges and make monitor electrically safe. (Model I)

HOME/HOBBY

Title	Author	Issue/Page	Debug	Annotation
Bout with the IRS, A	Blechman	9/80:58		How a TRS-80 can help in an IRS audit. (Model I)
CAL81	Strazzarino	12/80:128	4/81:22	Print out personalized calendar with graphics. (Model I)
Carpool	McCahan	5/80:62		Program to organize a carpool. (Model I)
Cold Comfort	Keen/Laughlin	10/80:176		Heating and insulation-analysis program. (Model I)
Decisions, Decisions	Walton	1/80:56		Basic program to aid in decision making. (Model I)
Duty Roster	Straw	3/80:127		Keep track of who's in charge of what for club or church functions. (Model I)
Fixer, The	Ashley	11/80:90		Program to keep photographer's darkroom organized. (Model I)
Graph Plotter	King	3/80:130		Plot how much you spend each month. (Model I)
Gregorian Converter	Borrmann	12/80:260		Convert Gregorian dates to Julian. (Model I)
Household Accountant	Andresen	2/80:114	4/80:27	Keep family expense records. (Model I)
IRS-80	McNeil	3/80:42		Keep tax records. (Model I)
Itinerary	Gorsky	4/80:95		Keep track of travel arrangements. (Model I)
Magazine Index	Klungle	4/80:114	2/81:31	Keep track of up to 300 articles in 16K. (Model I)
Merry TRSMAS	Taylor	1/80:32		Display Christmas greetings on TRS-80. (Model I)
Mind Your As and Ps	Leonard	9/80:174		Compile grocery lists in 16K. (Model I)
Model Conversion	Blackburn	5/80:128		Work out model measurements. (Model I)
Passing the Plate	Riekers	2/80:70		Keep track of Sunday church donations. (Model I)
Reference Library Index	Morgan	9/80:146		Keep track of articles in 16K. (Model I)
Soundex Codes	Hodge	3/80:138		Class similar-sounding names together for easier genealogical research. (Model I)
Tout I	Wilson	1/80:90		Horse-racing analysis program. (Model I)
Video Titler	Rotzien	4/80:121		Generate captions for video productions. (Model I)
Your Personal Calendar	Colsher	11/80:132		Put important dates, etc. on your 16K one-disk system. (Model I)

PROGRAMMING TECHNIQUES

Title	Author	Issue/Page	Debug	Annotation
Adventures in Roseland	Joffe	6/80:62		See Graphics
Beyond Blackjack	Thorsen	1/80:60		Learning Level I Basic graphics techniques. (Model I)
Cheap Trills with T-Bug	Joffe	11/80:168		How to get sound in machine code. (Model I)
Cutting & Splicing Basic	Nottingham	5/80:134		Reorder programs via block movement.
Display Formatting	Joffe	6/80:134		Techniques for a tidy screen display. (Model I)
Doodle Bug	Bishop	6/80:72		See Graphics.

Title	Author	Issue/Page	Debug	Annotation
Free Format Search	Riekers	5/80:94		See Utilities.
Get Serious	Pape	10/80:93		Using the RST instruction. (Model I)
Hidden Codes/ Missing Chip	O'Connor	1/80:93		Use commands the RS manuals don't mention. (Model I)
INKEY$	Himler	4/80:103		Enhance computer/user interaction. (Model I)
Keyboard Interrogation	Yarbrough/Vesteen	6/80:140		Use PEEK to directly access keyboard memory. (Model I)
Long and the Short of It	Bole	11/80:180		Format RS Quick Printer output for screen dumps of game displays. (Model I)
Merry TRSMAS	Taylor	1/80:32		See Home/Hobby.
Multiple USRs	Ventimiglia	4/80:80		Get up to 10 USR subroutines. (Model I)
Music Maestro!	Pape	1/80:36		Machine-language and Basic tone-generating routines. (Model I)
Now It's Time for... Name That Routine	Cornell	12/80:160		Use mnemonics for Basic functions call, with Assembly-language listing. (Model I)
Of Two-Dimensional Arrays	Conhaim	11/80:160		How to use two-dimensional arrays. (Model I)
Randomness	Carpenter	6/80:65		Use bargraphs to see how RND works in a program. (Model I)
Rom Routines	Thielke	2/80:106	10/80:14	Use Microsoft ROMs for programming shortcuts. (Model I)
Smart Programs	Lovy	11/80:202		Write programs with self-modifying code. (Model I)
Soundex Codes	Hodge	3/80:138		See Home/Hobby
Stringy Machine Code	Grimes	9/80:152		String-packing techniques with Basic program. (Model I)
Super Graphics	Moyer	10/80:202		See Graphics.
U-Boat	Borrmann	8/80:68		See Games.
Useful USR(0) Function	Kepner	10/80:100		Ways to imbed and use USR(0) in programs. (Model I)
Variations on a Theme	Bullitt	9/80:188		Use Print to print lines. (Model I)

SCIENCE/MATH

Title	Author	Issue/Page	Debug	Annotation
Biorhythms	Holthausen	3/80:117		Get a numerical indication of daily biorhythms. (Model I)
Divine Proportions	Cecil	9/80:170		Generate figures to study aesthetics. (Model I)
Equations	Joffe	3/80:84		Algebraic equation solutions.
Genotype	Rauber	10/80:188		Program to estimate risk of having children with defects for parents with genetic problems. (Model I)
Linear Meter Design	Thibodeau	7/80:58		Program to calculate values for linear dials. (Model I)
Oh No! Calculus	Joffe	1/80:114		Calculus demystified.
Process Control	Hoffman	2/80:104		Prevent problems caused by over-adjustment of a process containing random errors. (Model I)
Real Roots	Daniels	7/80:166		Compute square roots in Level I Basic fast. (Model I)
Root Routines	Gerald	2/80:73		Solve algebraic equations. (Model I)
Telepathy	Warren	1/80:54		Computer-simulated Zener cards to test for ESP. (Model I)

TUTORIALS

Title	Author	Issue/Page	Debug	Annotation
Article Called Intrepid	Gorsky	9/80:178		How coding works—cryptography on the computer. (Model I)
Assembly-language Trainer	Colsher	6/80:118		Assembly-language program that shows the result of most machine instructions. (Any Z80 computer)
Basic Switchyard, The	Perkins	7/80:52		How the Basic interpreter works.
Beginner's Formatting	Keller	7/80:108		Put graphics where you want them. (Model I)
BINAX KIBUFF	Blair	9/80:187		ROM routines not listed in your manual. (Model I)
Break Disable	Rastin	4/80:128	6/80:12	Software to prevent unwanted breaks. (Model I)
Business Programming	Clarke	5/80:78		How to write a business program.
Cassette Problems	Stoner/Barker	1/80:62		Reasons why a cassette won't load and possible solutions. (Model I)
Disk Files	O'Brien	7/80:88		Use random and sequential access on your disk system. (Model I)

Title	Author	Issue/Page	Debug	Annotation
Disk Primer, A	O'Brien	1/80:130		How disk drives work and what they can do.
EDTASM on Disk	Butler	1/80:120	5/80:28	Convert RS E/A from tape to disk. (Model I)
Equations	Joffe	3/80:84		See Science/Math.
Floppy PIMS	Herman	2/80:80		Convert SCELBI Personal Information Management System to run on Disk Basic. (Model I)
Get T-Bug High	Rappaport	1/80:118	3/80:13	Move T-Bug to higher memory. (Model I)
Graphic Sort	de Zoysa	8/80:166		Graphic depiction of how the computer sorts. (Model I)
Inside Level I	Meushaw	6/80:96		Make better use of Level I. (Model I)
Inside the ROMs	Stock	3/80:94		Using built-in Assembly-language subroutines. (Model I)
Inside- out Debugging	Ogren	11/80:222		Graphics techniques to get game figures on screen. (Model I)
Into the 80s, Pt. I	Sinclair	9/80:50		Pointers for beginners. (Model I)
Into the 80s, Pt. II	Sinclair	10/80:68		String variables, Clear command, Input tips. (Model I)
Into the 80s, Pt. III	Sinclair	11/80:70		If. . . Then. . . Else, Clearing methods, and more explained. (Model I)
Into the 80s, Pt. IV	Sinclair	12/80:82		Storing variables, dimensioning, loading strings, matrices, cutting strings, and more. (Model I)
Keyboard Information	Levy	1/80:138		Use INKEY$ more efficiently. (Model I)
Machine-code USR	McDonald	8/80:178		Incorporate machine code into Basic programs. (Model I)
Manipulative Wizard, A	Adams	12/80:94		How to handle array management. (Model I)
Modification Update	Richards	7/80:84		Things you can do to modify your computer. (Model I)
My Way	Meushaw	9/80:138		Self-help routines to let the computer help solve your computing problems. (Model I)
Mysteries of Level II ROM	Griswold	12/80:147	4/81:22	Jump addresses, compression codes, variable storage formats, etc. of Level II ROMs. (Model I)
"Next" Trap, The	Borrmann	9/80:208	11/80:22	Avoid the ?NF error message. (Model I)
Oh No! Calculus	Joffe	1/80:114		See Science/Math.
Pascal I & II	Monsour	5/80:38		A look at two commercial Pascal interpreters. (Model I)
Printer Calibration	Rexrode	9/80:94		Ways to ensure your labels or paper is properly aligned before you print.
Pulling Strings Together II	Adams	10/80:76		A look at string management instructions. (Model I)
Pulling Strings Together I	Adams	9/80:62		How to manipulate string data. (Model I)
Punch Out Your Disks	Taylor	10/82:140		Get double-sided disks with a paper punch.
Relocate with PEEK POKE	Rappaport	2/80:88		Move machine-code programs aound using a Basic program. (Model I)
Saving Money	Acres	7/80:110		Buy mail order and install yourself.
Smart Programs	Lovy	11/80:202		See Programming Techniques.
Towards Machine Language	Joffe	8/80:144	12/80:18	How to begin learning machine language.

UTILITIES

Title	Author	Issue/Page	Debug	Annotation
Append It!	Gerald	2/80:82		Add Basic statements to your program from cassette. (Model I)
Assemble It Yourself	Koch	12/80:212		Modify EDTASM to run without disk or cassette, and to use the editor as a WP. (Model I)
AutoPOKE	Kump	8/80:132		Load machine-language routines with a Basic routine. (Model I)
Babybeep	Kitsz	4/80:68		Machine-language sound routine. (Model I)
Babybug I	Kitsz	2/80:42	4/80:27	Machine-code monitor. (Model I)
Babyroot	Kitsz	5/80:76		Find defective memory locations. (Model I)
Backup/Display	Lindley	5/80:96		Assembly-language program to back up tapes and display contents in ASCII and hex. (Model I)
Basic Basic Renumbering	Orleff	1/80:82		150-byte renumbering program. (Model I)
Basic Terminal	Noreault	4/80:133		Turn the TRS-80 into an intelligent terminal. (Model I)

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Basic Word Processor	Hinrichs	5/80:50		WP in Basic. (Model I)
Beyond Shell Metzner	Walker	9/80:192		Basic sort routine. (Model I)
Buffer Analysis	Chambers	6/80:132		Display buffer contents in hex, ASCII, or decimal. (Model I)
Cassette File	Tallman	8/80:174		Use the first program on the tape to keep track of the others. (Model I)
CLOAD Assembly Language	Baker	6/80:146		CLOAD Assembly-language Level I programs. (Model I)
CLOAD Machine Language	Schimelman	2/80:68		Modify your machine-language monitor. (Model I)
CLOAD Monitor	Whaland	5/80:86		Get a graphic display when CLOADing Level I or II. (Model I)
Compac	Romanchik	12:80:198		Assembly-language subroutines to speed up I/O programming. (Model I)
Competition's Cursor, The	Bishop	9/80:210		Get a flashing cursor in Level II Basic. (Model I)
Compress It!	Powers	2/80:118		Take out remark statements. (Model I)
Cross Reference	Camp	11/80:128		Basic program that builds cross reference of machine-language calls and jumps. (Model I)
Custom EDTASM	Blair	8/80:122	11/80:22 1/81:28	Assembly-language printer driver for EDTASM. (Model I)
Decwriter Driver	Beauchamp	6/80:111		I/O driver to interface DEC LA-34 terminal to a TRS-80, in Assembly language. (Model I)
Deflower Your Debug	Walter	9/80:196		Access previously inaccessible memory locations. (Model II)
Delay Loop	Joffe	9/80:168		Delay or timing loops for Assembly-language programs. (Model I)
Disk File Protection	Keen/Dischert	8/80:164		Protect disk files from being overwritten. (Model I)
Disk Index	Cheshire	2/80:124		Keep list of programs on file. (Model I)
Displaced Programs	Moehlis	7/80:158		Move machine-language programs to high memory and save them to disk. (Model I)
Document Those Variables	Noel	9/80:88	2/81:12	Program that lists variables and tells what they represent. (Model I)
Doodle Bug	Bishop	6/80:72		See Graphics.
DOS Machine-code Loading Techniques	Turner	11/80:208		Keep routines in high memory from conflicting with each other. (Model I)
Dvorak Keyboard, The	Boyd	12/80:66		Program to convert your keyboard from Qwerty to Dvorak format. (Model I)
Etch-A-Screen	Shrum	5/80:116		Design video layouts for use in programs. (Model I)
Extra Errors	Moses	3/80:125		Add error messages. (Model I)
Extra Variables	Clark	1/80:134		Get extra variables in Level I. (Model I)
FASTDOS	Neher	5/80:114		Modify TRSDOS for more speed. (Model I)
Format 40	Adams	7/80:162	10/80:14	Basic printout format routine. (Model I)
Fractional Input	Cecil	5/80:142		Use fractions instead of decimals in input statements. (Model I)
Free Format Search	Rickers	5/80:94		Search a data base character by character. (Model I)
Free Space	Cornell	9/80:68		Load this utility that deletes spaces, line feeds, and remarks into the input buffer. (Model I)
Graphics Coder, The	Racine	8/80:160		Generate a video-display worksheet to compute proper graphics POKE locations. (Model I)
Heathkit Interface	Kunk	7/80:114		Increase the speed of the Heathkit H14 printer with software. (Model I)
Hex Display	Campbell	6/80:88		Convert hex to decimal. (Model I)
Holiday Cheer	Kerr	12/80:132	4/81:22	Address list and letter-writing program in Basic. (Model I)
Input with Insight	Decker	10/80:138		Get around the limitations of the Input command. (Model I)
Invisible Password	Conley	8/80:107		Hide your passwords on disk systems with NEWDOS. (Model I)
KBFIX Fix	Andreasen	4/80:108		Move KBFIX in memory with a Basic program. (Model I)

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Keyword List Plus	Decker	12/80:263		Routine to print keyword table with starting addresses (Model I)
KWIC Index	Sparks	3/80:60	5/80:28 6/80:12	Find articles with this technique. (Model I)
Less Is More	Winterbauer	12/80:259		Short Assembly-language print routine. (Model I)
Level II to Level I	Wolf	4/80:62		Convert Level II object tapes to Level I CLOADable format. (Model I)
Listen to Your Keyboard	Domuret	1/80:109		Debounce program with audio feedback on keystrokes. (Model I)
Lowercase with Strings Attached	Chepko	8/80:121		Program to generate lowercase. (Model I)
LPRINT Formatter	McCormick	2/80:120		Format printouts. (Model I)
LPRINT Routines	Werner	3/80:105		Print out data from the screen. (Model I)
LPVIDEO	Powers	5/80:136		Assembly-language program to print out screen contents. (Model I)
MACROPOKE Monitor	Suter	4/80:90		Incorporate machine-code routines in Basic programs. (Model I)
Memory Dump	Joffe	5/80:144		Look at memory via PEEK. (Model I)
Memory Sizer	Decker	10/80:114		Set memory size from Basic. (Model I)
Menu List Selection Subroutine	Rowlett	11/80:195		Select menu items by number. (Model I)
Mix Your Own PIMS	Busch	11/80:212		Add a directory to Scelbi's Personal Information Management System. (Model I)
Music Maestro!	Pape	1/80:36	4/80:27	See Programming Techniques.
New Restored	Fordham	1/80:84		Recover programs after typing New and restore to anywhere you like with these Assembly-language routines. (Model I)
Pencil RS-232 Driver	Kinsey	8/80:170		Use Electric Pencil and a printer through the RS-232C interface. (Model I)
Position Display	Frost	9/80:150		Display titles in programs or have a message stand out, draw borders. (Model I)
Pow—Bang—Zap—(Crash)	Brandolini	11/80:230		Assembly-language sound routine that uses cassette I/O port. (Model I)
Progdata	Kelley	5/80:126		Edit Assembly/Basic programs. (Model I)
QWIKDISK	Nazarian	9/80:206		Software patch to cut track access time in half. (Model I)
Relocate with PEEK POKE	Rappaport	2/80:88		See Tutorials.
Restore Data Pointer Control	Cecil	12/80:257		Set data pointer to a place other than the start of the data list. (Model I)
Resurrect It!	Quindry	11/80:206		Recover Basic programs lost due to errors in Basic or object-code programming. (Model I)
Scatterplot	Genovese	7/80:128		Use the Pearson Product-Moment correlation to examine one variable as a function of another. (Model I)
Screen Editor	Colsher	3/80:122		Document your programs in Level I. (Model I)
Screenprint	Frankenberg	5/80:152	8/80:16 9/80:14	Screendump by pressing I, J, and K simultaneously. (Model I)
Service Tape	Flatley	4/80:106		Load a collection of utility programs with one command. (Model I)
Slash Zero	Richardson	8/80:108		Assembly-language routine to print out all zeros with a slash. (Model I)
Slow Scroll	Lewis	9/80:202		Video driver mod to slow scrolling. (Model I)
Smart Terminal	Shirley	1/80:104		Assembly-language program to make your TRS-80 an intelligent terminal for a larger system. (Model I)
Software Lock	Kelleher	8/80:118		Limit access to machine-code routines. (Model I)
Sort 80K in 6K!	Fitchhorn	1/80:102	3/80:13	Use disk random access to sort more data than fits in memory. (Model I)
Sound X	Baker	7/80:136		Machine-language sound routine. (Model I)
Spool and Despool	Gentry	3/80:46		Print files while running another program. (Model I)
Super Sound	Morr	5/80:130		Assembly-language sound-generating routine using cassette recorder. (Model I)

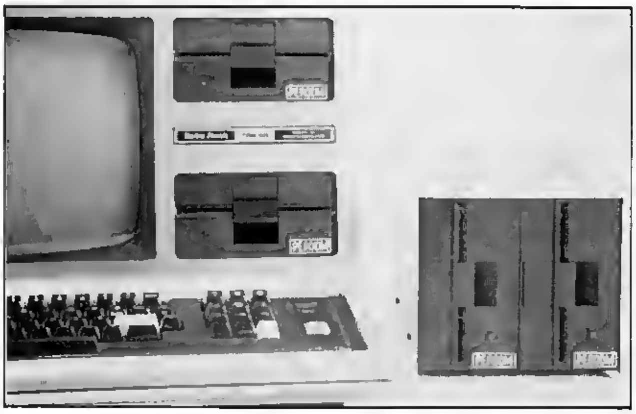
Title	Author	Issue/Page	Debug	Annotation
T-Bug and Then Some	Paxton	11/80:172		Add Clear, Dump, and Shift commands to T-Bug. (Model I)
T-Bug for II	Curtis	4/80:84		Shorten Level II T-Bug and customize it. (Model I)
Take Me Beyond Your Leader	McTernan	10/80:210		Move cassette past leader to oxide. (Model I)
Tape Librarian	Herold	8/80:100		Keep track of your tapes. (Model I)
TCopy	Stevens	7/80:160		Copy tapes with source or object code in one step. (Model I)
Test Your Memory	Chepko	3/80:80		Test program for new memory chips. (Model I)
Tinycomp	Bohlke	5/80:44		Basic program to compile into Assembly language. (Model I)
Triple Play for T-Bug	Johnson	10/80:207	5/80:22	Put T-Bug into three different memory locations. (Model I)
T Tape	Stevens	1/80:122		Analyze and adjust cassettes for better CLOADS. (Model I)
Uni-key	Archer	9/80:76		Enter Basic keywords in one stroke. (Model I)
Up and Down	Parris	11/80:177		Rerecord any 500-baud tape without fumbling with sound levels. (Model I)
Variable Scroll	Colsher	10/80:134		Get a screen display of variables used at any time. (Model I)
Versatile Input	Wilde	9/80:98		Routine for easy, elegant data input. (Model I)
Walking Words	Borrmann	9/80:173		Move words across the screen, left to right. (Model I)
Whazit?	Penny	3/80:115		Identify System and Basic tapes. (Model I)
White	Commander	7/80:94		Command to fool interpreter into executing your own commands. (Model I)
Winking Cursor	Lovy	1/80:68		Give input statements a winking cursor or graphics block. (Model I)
You Can Call It... Ray	Kornfeld	11/80:226		Read disk directories, number programs, and run them in one keystroke. (Model I)

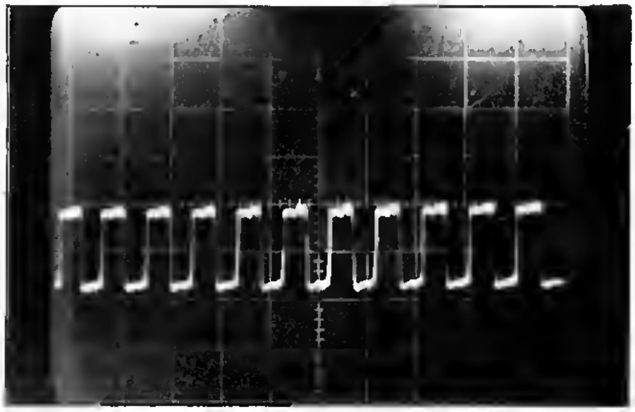
1981

BUSINESS

Title	Author	Issue/Page	Debug	Annotation
After the Goldrush	Frost	1/81:120		Calculate value of your precious metals. (Model I)
Analytic Inventory Management	Harper	12/81:314		Organize your inventory management. (Model I)
By Appointment Only	Busch	3/81:152		Appointment-tracking program. (Model I)
Dollars and Sense	Andrews	5/81:216		See Utilities.
Exponential Forecasting	Gorney	4/81:266		Forecast demand for your products. (Model I)
General Ledger	Conhaim	5/81:222	7/81:20	General ledger program for 32K and disks. (Model I)
House Detective, The	Cominio	11/81:160		Real estate information library. (Model I)
How to Buy & Sell Houses	Whitman	5/81:103		Program to figure wraparound mortgages. (Model I)
Income Averaging—1980	Grothman	3/81:136		Program to aid in figuring your taxes with income averaging. (Model I)
Investment du Jour	Honess	11/81:190	3/82:18	A look at the basic aspects of a business package.
Investment Property Analysis	Sparks	11/81:144		Estimate income from rental property. (Model I)
Landlord	Tuohy	2/81:154		Evaluate potential real estate investments. (Model I)
Line Up!	Anderson	3/81:174		Simulate waiting lines to study check-out counter efficiency. (Model I)
Loan Wrangler	Jensen	3/81:240		Loan calculator program. (Model I)
Micros—Business or Pleasure	Latamore	7/81:90		How Tandy is pushing the TRS-80 as a business machine.
Mind Your Own Business	Glau	3/81:200		Assembly time-cost analysis program. (Model I)
Paper Mountain	Wierenga	11/81:118		See Education.
Pacemaker, The	Tinis	11/81:220		Assembly time-cost analysis program. (Model I)

Title	Author	Issue/Page	Debug	Annotation
Punch Out	Conroy	2/81:198		Program to aid tallying of time cards. (Model I)
Software Broker, The	Harper	6/81:268	10/81:19	Seven-program investment-analysis package. (Model I)
Some Fundamentals	Tune	2/81:216		Use the TRS-80 as an effective business computer. (Model I)
State of the Union	Reid	11/81:321		Financial trend monitor for credit unions. (Model I)
Take a Letter	Mullin	8/81:290		See Utilities.
Tenant Tracker	Kwascha	11/81:138	4/82:26	Computerize your rental records. (Model I)
Ups and Downs of Graphs	Foley	11/81:176		See Graphics.





EDUCATION

Title	Author	Issue/Page	Debug	Annotation
Alternate Course, Pt. I	Duffin	8/81:180	12/81:28	Programs to teach data processing and alleviate computerphobia. (Model I)
Alternate Course, Pt. II	Duffin	9/81:204		See Part I.
Alternate Course, Pt. III	Duffin	10/81:180	3/82:22	See Part I.
Alternate Course, Pt. IV	Duffin	11/81:274		See Part I.
Analogies	Spencer	9/81:328		Analogy-quiz program. (Model I)
Calling All Teachers	Hedden	11/81:306		Package for teachers' record keeping. (Model I)
Classroom Computing	Droegemueller/Bell	2/81:78		How a school started a computer-education course.
Classroom Doodles	Rosenberg	2/81:86		Graphic program to encourage problem solving and
Foreign-language	Robinson	5/81:219		Use Electric Pencil to build a foreign-language dictionary. (Model I)
Green Register, The	LaBorde	8/81:214		Program to keep student records. (Model I)
Math Derby	Duffin	9/81:244		Math game to teach the basics. (Model I)
Notes from the Classroom	von Kaenel	2/81:96		Tips on introducing computers to the classroom.
Paper Mountain	Wierenga	11/81:118		Program to keep school personnel records. (Model II)
Programming for Education, Pt. I	Weintraub	2/81:68		Things you should know about writing educational software.
Pogramming for Education, Pt. II	Weintraub	4/81:144		How to go about writing test programs.
Programs for the Handicapped	Nowak/Muswick	10/81:141		Programs handicapped children can use to learn. (Model I)
Project Local	Petrakos	2/81:74		CAI as it is done at one school.
Quiz	Olsen	8/81:188		Use the computer to test students. (Model I)
Rushing Toward Courseware	Petrakos	4/81:74		How computerized instruction is affecting the classroom.
Specific Heat	Fetchko	12/81:220		Program to help students learn how to find the specific heat in chemistry lab. (Model I)
Teacher Mod	Fish	9/81:272		Modify ISI's teacher program for networking. (Model I)
Vocabulary Builder	Zimmerman	2/81:226		Program to aid learning a foreign vocabulary. (Model I)
Walls of Jericho, The	McNichols	9/81:292		Teach the Bible with this program. (Model I)

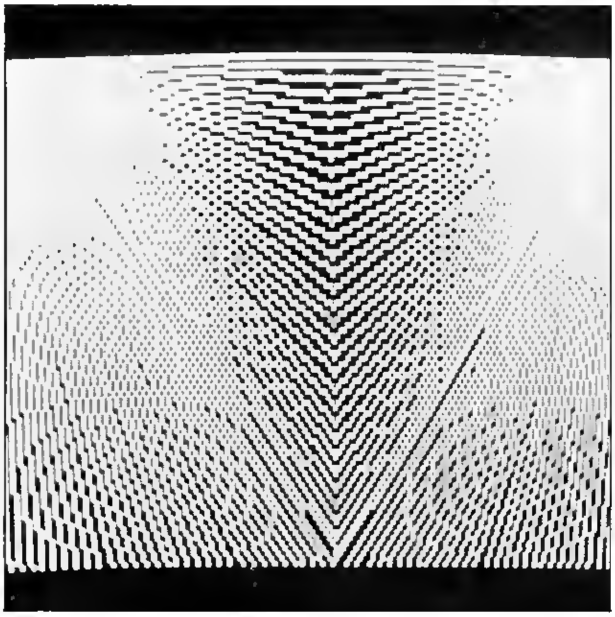
GAMES

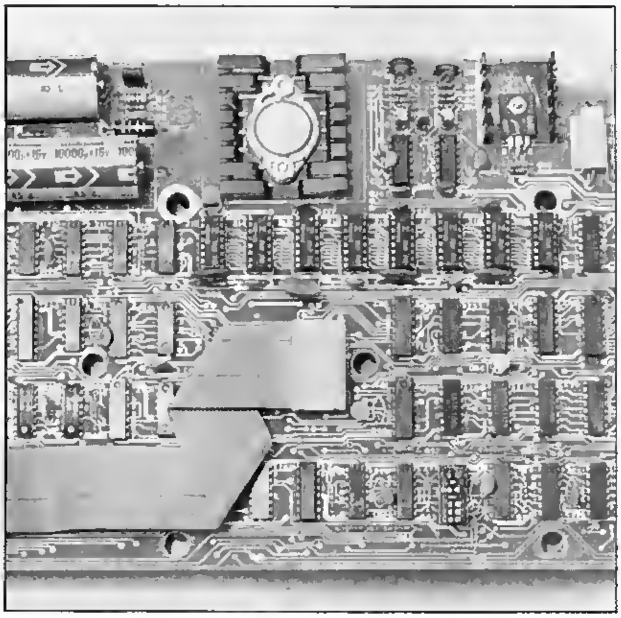
Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Boa	Myers	5/81:294		Two-player graphic game of survival. (Model I)
Capture the Computer	Fisher	12/81:176		Computer version of the game of Tag. (Model I)
Chess Tutor, The	Dowd	12/81:154	3/82:18 4/82:24	Let the computer teach you chess. (Model I)
Color Concentration	Wrye	12/81:298		Like the tv show. (CC)
Colorful Maneuvers	Wood	11/81:328		Outmaneuver a friend. (CC)
Compukala	Moller	3/81:278		Computer version of the game of Kala. (Model I)
Dessert Solitaire	Ratke	12/81:226		Based on the board game. (Model I)
Digits for Fun	Wirth	8/81:304		Math Puzzle. (Model I)
Dot Game, The	Welcher	4/81:210		Simple game to connect the dots. (Model I)
Formula 80	Sprague	8/81:122		Race sports cars on your screen. (Model I)
Frankenstein	Nicholas	8/81:92		Three word games in one. (Model I)
Let's Get Rude!	Ramella	12/81:142		Removal game with pejorative prompts. (Model I)
Level II Black Box	Jones	7/81:217		Find the contents of The Black Box. (Model I)
Lunar Lander Revisited	Beringer	6/81:244		Another version of an old standard. (Model I)
Magic Cube, The	York	12/81:346		Simulation of the Rubik's Cube. (Model I)
Magic Trick, The	Busch	9/81:300		Party game. (Model I)
Master Mind	Hope	10/81:122		Computer version of game, using AI techniques. (Model I)
Math Derby	Duffin	9/81:244		See Education.
Micro-basketball	Weindorf	3/81:118		Computer basketball game for 16K. (Model I)
Micro-Yahtzee	Johnson	10/81:302		Computer dice game. (Model I)
Simon	McGlumphy	8/81:112		Computer simulation of Simon in 4K. (Model I)
Simul-80—a Weird Game	Kitsz	4/81:154		Game that simulates what happens inside a microprocessor. (Model I)
Space Empires	Smith	8/81:106	5/82:26	Strategy space game. (Model I/III)
Space Potatoes	Brumme/Abel	8/81:102		Shoot down cosmic spuds. (Model I)
Star Colony	Beringer	10/81:334		Colonize and explore the star system. (Model I)
Star Guard	Beringer	8/81:116		Three-dimensional space game. (Model I)
Startrek 4.0	Commander	8/81:86	1/82:24	Classic Star Trek space game. (Model I/III)
Subdestroy	Cominio	6/81:186		Arcade-style battle game. (Model I)
Supermaze	Batie	6/81:160	9/81:16	Maze game, using arrays. (Model I)
Tank	Rice	8/81:104		Arcade game for preschoolers. (Model I)
This Ain't No Party!	Fason	4/81:179		Party game that interviews guests. (Model I)
Walk on the Monster Side	Latamore	8/81:84		Short history of games and game development.
Writers of a Lost Art	Albino	12/81:290		See Tutorials.

GENERAL

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
80 Interview with Kornfeld/Roach	Robertson	3/81:82		Two interviews, one with Lew Kornfeld and one with John Roach, both of Tandy.
80 Limited, The	Latamore	12/81:128		How a TRS-80 runs a model train setup.
Between the Lines	McTernan	3/81:216		Level II manual errata fixed, with thoughts on the rest of it.
Color Computer First Impression	Kilmon	8/81:286		Thoughts on the "new" computer.
Computerization/ Workplace	Brown	11/81:110		How the Computer Revolution is affecting the management/labor relationship.
Date Maker, The	Grosse	8/81:274		Julian-to-Gregorian date converter. (Model I)
Digital Delivery	Latamore	11/81:112		Future of electronic mail.
Dragnetwork	Kennedy	6/81:122		How a police department set up a network with microcomputers.
Field Guide to Computerists	Doherty	5/81:208		Humor piece on different types of computer users.
Getting Involved	Batty	7/81:179		Looking back on how one got started with micros.
It's All Robotese to Me	Gesamte	9/81:101		Humor—a robot glossary.
Just a Matter of Time	Busch	3/81:196		Thoughts on using a micro to save money around the home.
Lenny's Story	Busch	12/81:384		Humor piece on virtues of making back-ups.

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Queue Theory	Neibauer	12/81:368		Simulate a gas line. (Model I)
Realm of Science Fiction	Robertson	10/81:136		A look at AI in science fiction.
Unexpurgated Version, The	Busch	4/81:178		Humor piece on the "real" definitions of some computer terms.
User's Groups	staff	11/81:272		A list of TRS-80 user's groups.





GRAPHICS

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
80 Cartoon Capers	DeVigili	9/81:320	12/81:28	Generate cartoons on the screen. (Model I)
Advanced Graphics Techniques, Pt. I	Boothe	4/81:116		Use disk commands without disks to get graphic printouts. (Model I)
Advanced Graphics Techniques, Pt. II	Boothe	5/81:119		See Part I.
Advanced Graphics Techniques, Pt. III	Boothe	6/81:68		Manipulate three-dimensional objects on screen. (Model I)
Banner Banter	Vann	2/81:128		Print out large-letter banners on the Quick Printer (Model I)
Can Computing Be Art?	Conroy	2/81:122		Reproduce pictures with characters on a printer. (Model I)
Classroom Doodles	Rosenberg	2/81:86		See Education.
Color by Percom	Kalinowski	1/81:68		Use Percom's Electric Crayon to get color on the Model I.
Curve Plotter	Zimmerman/Stanley	9/81:256		See Science/Math.
Doodlebug	McKenna	1/81:208		Draw on the screen from the keyboard. (Model I)
Perspective on Cubes, A	Gerhardt	1/81:182		How to program computer to draw cubes. (Model I)
Picture This	Keen/Dischert	9/81:214		Animation via string packing. (Model I)
POKE A, Color Computer	Esposito	12/81:362		Access graphics modes in the CC.
Rotation	Yelling	9/81:154		See Programming Techniques.
Tiger with Dots, A	Somers	6/81:96		Control matrix print-head needles through software on the IDS 440. (Model I)
Title Graphics	Kalinowski	9/81:302		Get flashy titles on the screen for your programs. (Model I)
Turn of the Screw	Boothe	4/81:117		Three-dimensional, rotating graphics for printer and video. (Model I)
Unlocking the Graphic Code	Weintraub	6/81:147		Relationships between graphic elements and their ASCII character codes. (Model I)
Ups and Downs of Graphs	Foley	11/81:176		Generate graphs. (Model I)

HARDWARE

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
80 Appliance Control	Lewart	3/81:230		Build a circuit to control home appliances with software-generated tones. (Model I)
Alternate Source, An	Conner	5/81:254		Battery-charger-based power supply.
Another Noise Maker	Erb	12/81:312		Sound synthesizer. (Model I)
Audio Interface	Batie	1/81:172		Cassette dubbing, aural and video load monitoring, debounce, and more with this device. (Model I)
Baudot LPRINT	Rister	11/81:314	2/82:28	Interface the Teletype Model 15 to the TRS-80. (Model I)
Cruise Control	Waltjen/Placido	12/81:320		Auto-adjusting speed-up mod. (Model I)
Eye-80	Hawkes/Reese	6/81:140		Photo cell interface that will allow a Model I to detect objects and speed and direction.
Gold-plated 80, A	Martin	12/81:302		Gold plate your contacts.
Hang Person	Hilton	10/81:320		Hangman with sound. (Model I)
Hard and Soft Printware	Sinclair	10/81:234		Build a printer interface. (Model I)
Hardwire the RS-232	DeJarnette	5/81:202		Solder the RS-232 in place.
High-Density Graphics Interface, Pt. I	Murray/Fowler	4/81:134		Get 192-by-96 resolution on your monitor. (Model I)
High-Density Graphics Interface, Pt. II	Murray/Fowler	5/81:80	10/:81:19	See Part I.
Inverse Video	Smith	5/81:176		Build a device to print characters in black on white. (Model I)
Joysticks for the Model I	DiNunzio	6/81:157	10/81:23	Build a two-joystick interface.
Last CLOAD Fix, The	Stanley	7/81:171		Build your own X2X CLOAD fix. (Model I)
Lightning Strikes Twice	Fowler/Murray	7/81:184		Build a permanent transient suppressor.
Lower Cost Lowercase	Van Praag	4/81:228		Lowercase Mod and KB fix. (Model I)
Spare-time Generator, The	Mehesan	5/81:264		Quick sort utility for name lists. (Model I)
Split and Splice	Mills	9/81:228		Split up Basic programs, convert Level I to II, merge together again. (Model I)
Spooler, The	Gault	12/81:222	3/82:18	Use your computer while the printer prints. (Model I)
Superlist	Jones	11/81:333		Get more readable listings. (Model I)
T-Bug III	Kleinfelter	11/81:390		Convert Model I T-Bug to Model III.
Take a Letter	Mullin	8/81:290	12:81:26	Letter-writing program. (Model I)
Take T-Bug Higher	Stanley	5/81:284		Move T-Bug to high memory. (Model I)
Tale of Two Drivers, A	Blair	10/81:326		Printer patches for Level II Basic and NEWDOS. (Model I)
Terminal Plus	Gorsky	1/81:226		Assembly-language terminal program for 48K disk systems. (Model I)
Through-boot	Smith	10/81:312		Initial program loader for TRSDOS. (Model I)
TRS-80 Disassembler	Wuebker	8/81:240		Read bits in ROM and convert them to Z80 mnemonic instructions. (Model I)
UC/LC Fix	Thurlow	8/81:260		Get INKEY$ working properly with the RS lowercase mod. (Model I)
Underscoring Scripsit	Iseli	6/81:246		Underline in Scripsit. (Model I)
Variable Lister, The	Webster	7/81:259		List variables. (Model I)
Voice Synthesizer	Hall	10/81:146		Program to generate words on the RS Voice Synthesizer. (Model I)
Weak Link	Denholtz	10/81:360		Strengthen NEWDOS80's Chain command. (Model I)
Where Have All the GOTOs Gone	Borrmann	3/81:236	6/81:16	Program that lists line numbers with specific instructions. (Model I)
Wherzit—Keyword Index	Fox	4/81:252	9/81:16	Indexing program that searches by title. (Model I)
Write Stuff, The	Barnes	12/81:390	3/82:22	Assembly-language word processor. (Model I)
ZBug...Super Debug Monitor	Harrell	1/81:130	4/81:22	An Assembly-language monitor for 16K. (Model I)

HOME/HOBBY

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Auto Mentor, The	Sparks	5/81:87	7/81:20 8/81:16	Program to aid in buying a car. (Model I)

Title	Author	Issue/Page	Debug	Annotation
Being of Sound Algorithm	Weinberg	8/81:136		Write your will on your TRS-80. (Model I)
Cadet—The Decision Maker	Albino	10/81:298		Program to aid in making decisions. (Model I)
City Accessibility Calculation	Todd	12/81:310		Find out which city is most convenient for all group members to meet in. (Model I)
COINS	Lloyd	12/81:134		Catalog your coin collection. (Model I)
Computer Cantos	Griffiths	3/81:154		Computer-generated poetry. (Model I)
Computerized Complaint	Gillig	5/81:276		Send out complaint letters via computer. (Model I)
Dancin! A Disco Primer	Modla	5/81:194		Generate disco dance steps on screen. (Model I)
Dollar Down, A	Martin	5/81:113		Evaluate financing plan for a new or used car. (Model I)
Endorse It	Welcher	4/81:214		Basic check-writing program. (Model I)
Equine Equation Evaluation	Herold	9/81:262		Horse-racing program. (Model I)
Family Relationships	Horwitz	8/81:192		Genealogy program. (Model I)
Fat City	Busch	10/81:274		Freezer inventory program. (Model I)
Final Notice, The	Atkins	2/81:200		Personal payment scheduling program. (Model I)
Firestream	Gille	10/81:316		Calculate correct water pressure for fire hoses. (Model I)
Hold That Pose	Ebert	8/81:248		Photo and slide-indexing program. (Model I)
Home Buyer's Helper, The	Stinson	5/81:99	4/82:26	Estimate what it will cost you to buy a home. (Model I)
Hoops	Cornwell	10/81:184		Program to keep basketball statistics. (Model I)
House Plant Index, The	Chipman	12/81:116		Keep the vital statistics of your house plants on the computer. (Model I)
Lifespan	Wailand	9/81:252	12/81:28	Program that estimates your lifespan based on your lifestyle. (Model I)
Loan Sharp	Atkins	10/81:338		Figure interest on the Pocket Computer.
Mileage Manager, The	Frink	6/81:236		Keep track of the gas consumption of your car. (Model I)
Morse Resource, Pt. I	Joffe	8/81:150		Software to help you send Morse code. (Model I)
Morse Resource, Pt. II	Joffe	9/81:130		See Part I.
Nag Analysis	Crosby	7/81:243		Horse-handicapping program. (PC)
Philatelist's Friend, The	Castor	12/81:108	5/82:26	Catalog your stamp collection. (Model I)
Pocket Full of Chips	Dolan	12/81:230		Several common calculation programs. (PC)
Procedures Pricing	Keller	12/81:252		Keep track of the cost of living. (Model I/III)
Quick Riff on Synthesizer	Keen/Dischert	7/81:294		ARPs, Moogs, and TRS-80s.
Real Rules of 78s, The	Conhaim	7/81:289		Figure early loan repayment. (Model I)
Rule of 78s, The	Conhaim	6/81:296		Figure rates on early payment of loans. (Model I)
Runner's Logbook, The	Vose	12/81:100		Keep a log of running times and distances. (Model I/III)
Second Sourcing	Tinis	11/81:348		See Hardware.
Secret Ballot	Busch	9/81:286		Use your computer as a ballot box. (Model I)
Sharp Marketing	Atkins	12/81:150		Supermarket program. (PC)
Shoplist	Smith	3/81:274		Shopping-list program. (Model I)
Shopper's Aid	Maninger	12/81:172		Shoplist program. (PC)
Two Cents Worth	Clayton	4/81:192		Program to evaluate old pennies. (Model I)
Watt's It All About?	Hubert	3/81:219	6/81:16	Home electrical utility monitor program. (Model I)
Worldly Goods	Fail	10/81:220		Keep inventory of your belongings. (Model I)

PROGRAMMING TECHNIQUES

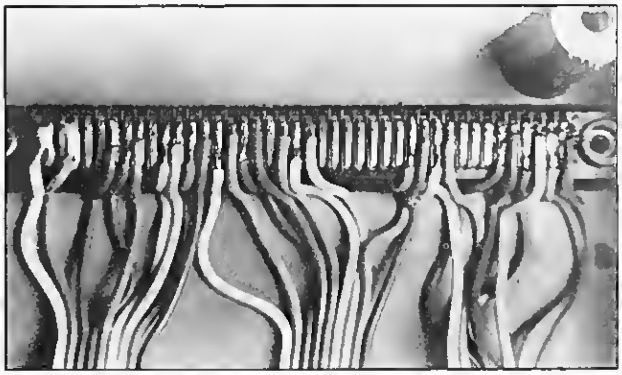
Title	Author	Issue/Page	Debug	Annotation
All About Program Files	Barnard	10/81:258		Use program files with small DBMs on cassette. (Model I)
All About Sorts, Pt. I	Gorney	8/81:208		See Tutorials.
Bascal	Metzler	9/81:334		Get a hybrid Basic/Pascal programming language.
Be a Super USR!	Alford	8/81:254		Ways to use USR calls. (Model I)
Chain Command Implementation	Pape	11/81:250		Program chain and edit programs and continue execution. (Model I)
Debug with GOTO	Pape	10/81:249		Save what's in memory when your program crashes.
Delete and Save	Langston	9/81:240		Delete lines and stay in the program. (Model I)

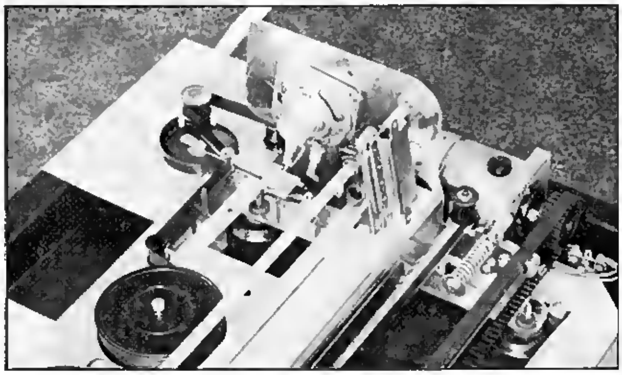
Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Do Not Enter	Weintraub	3/81:104		Use INKEY$ and eliminate prompts in programs. (Model I)
Doing Two Things at Once	Gorsky	3/81:178		Using interrupts in your programs. (Model I)
Efficient Cassette I/O	Sabin	1/81:218		How to better handle cassette data files. (Model I)
Erudite Arrays	Barnes	9/81:226		Use one array as a series of arrays in Level I. (Model I)
FOR My NEXT Trick . . .	MacLean	12/81:380		See Tutorials.
Formatted Screen Input	Byrnes	10/81:350		Easier way for data entry. (Model I)
Get Organized	Lesley	12/81:398		Organize your programming methods.
High-speed Sorts	Robson	9/81:220		Sort in machine language. (Model I)
How to Handle Those Random Files	Knecht	7/81:271		See Tutorials.
In Praise of Outlines	Drew	11/81:204		Outline your programs before you write them.
Loops and Arrays	Wein	4/81:246		Combine loops and arrays in your programs.
Multi-statement	Keen/Dischert	10/81:296		Using multi-statement lines in programs.
New Words for Basic	Neibauer	9/81:170		Simulate commands not in Level II Basic.
Not-So-Random Numbers	Zeigler	5/81:246		Generate random sequences using Poisson, exponential, or normal distributions. (Model I)
On Embedding Data	Adams	6/81:294		Tips on using Read. . . Data. . . Restore.
Pascal Dream, The	Krutch	6/81:174		Pascal as compared to Basic.
Perspective on Cubes, A	Gerhardt	1/81:182		See Graphics.
Picture This	Keen/Dischert	9/81:214		See Graphics.
Plan of the Page, The	MacLean	1/81:222		Improve your video page layouts for Level I. (Model I)
POKE A, Color Computer	Esposito	12/81:362		See Graphics.
Program Chaining and Local Variable Definitions in Basic	Brown	6/81:255	10/81:22	Chaining, variable storage and preservation, and string storage tips.
Programming with Voice Synthesizers	Werner	8/81:208		See Tutorials.
RAM Squeeze	Adams	8/81:230		Space-saving methods. (Model I)
Regression and Correlation	Honess	7/81:254		Using regression and correlation for forecasting.
Rotation	Yellin	9/81:154		How to rotate 3-D figures. (Model I)
Simple Syllogisms	Hoffman	10/81:132		AI techniques for theorem-proving programs. (Model I)
Slice & Dice Basic	Schneider	9/81:322		Write self-modifying Basic code.
Thoughts on For. . . Next	Amyx	10/81:354		Ways to use the loop.
Three's Not a Crowd	Mawdsley	8/81:292		Program with utilities in memory. (Model I)
Title Graphics	Kalinowski	9/81:302		See Graphics.
Undocumented Instructions	Cameron	7/81:296		Split sometimes unused 16-bit index registers in half. (Model I)
USR Usery	Freese	12/81:248		Put Variables and $ in USR calls. (Model I)
Vital Statistics	Honess	8/81:194	12/81:26	Build a program to summarize and display descriptive data value characteristics. (Model I)
Wild Glitch Hunting	Warren	3/81:186		Finding the more subtle bugs in your program.

SCIENCE/MATH

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
And Now Artificial Intelligence	Nelson	10/81:108		Some techniques used in AI research.
Applications in Real Time	Genet	6/81:127		How a Model I runs the night shift at an observatory.
Artificial Intelligence at MIT	Vose	10/81:118		A look at current research.
Artificial Intelligence—Technology and the Search for Self	Brown	10/81:103		State of the art of artificial intelligence research.
Curve Plotter	Zimmerman/Stanley	9/81:256		Graph math functions. (Model I)
Digits for Fun	Pirth	8/81:304		See Games.
Dome Time	Nickell	5/81:182		Analysis program for geodesic dome design.

Title	Author	Issue/Page	Debug	Annotation
				(Model I)
Function Plotter	Graue	2/81:126		Graph any function. (Model I)
Molality Crunching	Suder	2/81:194		Use atomic masses in practical problems on the computer. (Model I)
Number Cruncher	Barbarello	1/81:232		Data reduction program for statistical studies. (Model I)
On the Average	Gorney	5/81:292		Program to figure mean, midrange, mode values in averaging. (Model I)
Pocket Stats	Atkins	11/81:392		Median-length statistics program. (PC)
Population Estimation	Solomon	2/81:152		Analysis program based on Lincoln-Peterson techniques. (Model I)
Sheer Lunacy	Harris	12/81:386		Get a graphic representation of the moon for any point in time. (Model I)
Simple Syllogisms	Hoffman	10/81:132		See Programming Techniques.
Solar Altitude Plotter	Rea	8/81:294		Track the Sun's position. (Model I)
Specific Heat	Fetchko	12/81:220		See Education.
Sunrise. . .Sunset	Skramstad	10/81:272		Calculate time and place of sunset/sunrise. (Model I)





TUTORIALS

Title	Author	Issue/Page	Debug	Annotation
Adventures in Modemland	Blechman	10/81:264		Using TRS-80 with a modem. (Model I)
All About Program Files	Barnard	10/81:258		See Programming Techniques.
All About Sorts, Pt. I	Gorney	8/81:208	12/81:28 1/82:24	Choosing the correct sort method. (Model I)
All About Sorts, Pt. II	Gorney	9/81:148	1/82:24	See Part I.
Bascal	Metzler	9/81:334		See Programming Techniques.
Basic Difference, The	Blechman	10/81:290		Converting Model I programs to Model III.
Be a Super USR!	Alford	8/81:254		See Programming Techniques.
Be a Super User	Joffe	2/81:166		How to use USR. (Model I)
Better Documentation	Ferber	3/81:209		How to develop a trainer module for program users. (Model I)
Buyer's Guide to Printers	staff	6/81:84		Available printers compared.
Buyer's Guide: Peripheral	staff	12/81:183		What's available in peripherals for the TRS-80.
Clear N	Salsbury	9/81:312		What Clear N does. (Model I)
CLOAD Is Just a Five-Letter Word	Kitsz	1/81:114		Problems and solutions to CLOAD woes. (Model I)
Cobol: Ready and Waiting	Bradley	7/81:116		An introduction to Cobol.
Color Computer, The	Martel/Nicholas	6/81:202	9/81:16	A look at the CC's capabilities.
Coming to Terms	Fugate	7/81:112		Some common computer buzzwords defined.
Compile, Interpret, Assemble	Faulk	2/81:224		How the compiler, interpreter, and assembler work.
Conversion, The	Woeger	12/81:324		Hands-on approach to converting from a Basic to Assembly-language program.
Cost-effective Word Processor	Hewin	4/81:222		Putting together a cheap word-processing system.

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Datacom Decisions	Derfler	8/81:262		How to start your data communications system.
Disk DOSsier	Commander	3/81:68		What a DOS is; how TRSDOS works.
Disk to Tape	Hocking	3/81:250		Thought about using cassettes after getting disks.
Fast Clock DOS	Domuret	4/81:240		Speed up your DOS when you speed up your hardware. (Model I)
Fast Round-up, A	McComb	11/81:184		Choosing a word processor.
First Look at Forth, A	Krutch	7/81:162		Introduction to Forth.
Floppy Tales	Keener	2/81:220		How to use the Extaron Stringy Floppy for data storage. (Model I)
FOR My NEXT Trick . . .	MacLean	12/81:380		Uses of the For. . . Next loop.
Freebie, The	Adams	12/81:304		How to take 150 bytes out of reserved RAM on cassette systems. (Model I)
Going Pro	Moss	9/81:242		Tips on programming for money.
How to Handle Those Random Files		Knecht	7/81:271	Make efficient use of random-access disk files. (Model I)
Idiot's Guide to AL, Pt. I	Montgomery	5/81:168		Beginner's guide to Assembly-language programming. (Model I)
Idiot's Guide to AL, Pt. II	Montgomery	6/81:112		See Part I.
In Command	Rutledge	11/81:300		Stay in the Command mode. (Model I)
In Praise of Outlines	Drew	11/81:204		See Programming Techniques.
Ins and Outs of Edit, The	Schrader	12/81:358		Notes on using the Edit utility. (Model I)
Into the 80s, Pt V	Sinclair	1/81:100		Using math on the computer. (Model I)
Into the 80s, Pt. VI	Sinclair	2/81:100		Producing a menu, using subroutines, and more. (Model I)
Into the 80s, Pt. VII	Sinclair	3/81:92		Program construction, using variables, PEEK and POKE, and more. (Model I)
Join the Pascal Parade	Grothman	7/81:146		Introduction to Pascal.
Keep it in the Black	Johnston	5/81:188		Reink printer ribbons.
Keyboard Incantations	Robnett	11/81:352		Mysteries of the TRS-80 keyboard. (Model I)
Language Quest '81	Vose	7/81:96		What a computer language is and how they work.
Level II Utilities/Mod III	Bigelow	9/81:182		Modify machine-language subroutines for the Model III.
LLIST for Level I	Ogden	7/81:251		How to get Level I printouts from Level II. (Model I)
Macroprocessor for Basic	Olmstead	8/81:156		Using macros in logical programming.
Macroprocessor for Basic Pt. III	Olmstead	9/81:142		Why most compilers fail, and how to change that.
Macroprocessor for Basic Pt. IV	Olmstead	10/81:228		What the Metabasic compiler is.
Macroprocessor for Basic Pt. V	Olmstead	11/81:243		See Part IV.
Macroprocessor for Basic Pt. VI	Olmstead	12/81:206		See Part IV.
Memories Are Made of This	Randall	5/81:146	9/81:16	RAM/ROM explored with memory test routines. (Model I)
Model II Q and A	Yager	6/81:197		Common questions about Model II answered.
Modify Tiny Pascal for Disk	Harrell	7/81:154	12/81:26	How to make changes to put RS Tiny Pascal on disk. (Model I)
On Modems	Brown	1/81:98		Basic information on modems.
Pilot—Language of CAI	Hawkins	7/81:122		Introduction to Pilot.
Programming for Education	Weintraub	2/81:68		See Education.
Programming with Voice Synthesizers	Werner	8/81:208		Techniques for programming voices. (Model I)
Random Tricks	Perkins	2/81:168		Why Z80 random number sequences are somewhat predictable.
Regression and Correlation	Honess	7/81:254		See Programming Techniques.
Shadow Knows, The	Romkey	3/81:244		What some of the undocumented RAM areas hold. (Model I)
Short and Sweet	Gundlach	3/81:272		One line instructions that are useful. (Model I)
Smart Answers	Williams	4/81:298	3/82:24	Answer computer prompts in hex. (Model I)

Title	Author	Issue/Page	Debug	Annotation
Soft Characters	Moulton	10/81:285		Get six more characters from the RS generator ROM. (Model I)
Speedset	Winings	5/81:258		Calibrate your drive speed.
To Err Is. . . Forbidden, Pt. I	Adams	6/81:108		What the error messages the TRS-80 gives mean. (Model I)
To Err Is. . . Forbidden, Pt. II	Adams	7/81:291		How to use error traps.
USR Usery	Freese	12/81:248		See Programming Techniques.
Writers of a Lost Art	Albino	12/81:290		Write your own adventure games.

UTILITIES

Title	Author	Issue/Page	Debug	Annotation
80 Black Book	McGlumphy	12/81:278		Find ROM addresses fast. (Model I)
80 to RS-232 to TI-810	Herman	8/81:164		Software to interface the TI-810 printer to the TRS-80 via the RS-232. (Model I)
Auto Edit	Rollins	2/81:144	5/81:22	Find out where the current line pointer is located and simplify editing. (Model I)
Auto-key	Ghan	10/81:242		Define strings in Basic and execute USR commands to enter character from keyboard. (Model I)
Babydub	Kitsz	3/81:210		Copy most cassette programs under 16K with greater volume variation. (Model I)
Base Conversions	Yelvington	9/81:186		Convert number bases in Assembly language. (Model I)
Basic Disassembler, A	Delfine	7/81:244	2/82:28	Generate Assembly listings of machine code and examine Level II PROMs. (Model I)
Basic Shorthand	Radin	8/81:282		Enter Basic keywords with one keystroke. (Model I)
Basic—Enhanced Again	Goodwin	11/81:384		Twelve new commands for Level II Basic. (Model I)
Block That Cursor	Balewski	4/81:297	7/81:20	Turn the winking cursor into a block cursor. (Model I)
Building Bridges	Mueller	10/81:306		Move AIDS III's fields to or from Special Delivery's fields. (Model I)
Captran	Gorsky	6/81:192		Convert uppercase text files to upper/lowercase. (Model I)
Cheater Poker	Davies	12/81:356		Data-generating utility that is easy to use. (Model I)
Check Writer	Tzinberg	4/81:261		Patch for check-writing programs to produce professional-looking results. (Model I)
Clock Boot	Boehmke	2/81:112		Put the TRSDOS clock back on time after a reboot. (Model I)
Command File Modifier	Crawford	8/81:278		Create new command files or modify existing ones. (Model I)
Compare	Everett	4/81:271		Double check accuracy of ASCII disk dumps. (Model I)
Comprs	Barker	5/81:270		Machine-language routine to remove remarks and spaces from Basic programs. (Model I)
Constant Alternatives	Hand	3/81:225		Use decimal, hex, and octal constants in Level II Basic. (Model I)
Copyit	Balewski	10/81:370		Eliminate retyping duplicate data statements in programs. (Model I)
Copykill	Kelly	6/81:224		Copy or kill dozens of programs with one command. (Model I)
Crossdos	Means	10/81:288		Convert TRSDOS to CP/M. (Model I)
Crossref	Ewart	1/81:212		Get a variable cross-reference listing. (Model I)
Customized Commands	Rupert	11/81:292		Add merge and other commands to Level II Basic. (Model I)
Datafix	Barnard	10/81:358		Translate two or more keypad periods into a word or other punctuation. (Model I)
Datagen	Lewart	8/81:168		Get data statements from machine language to Basic. (Model I)
Dateline 80	Press	9/81:333		Create Basic data statements from object-code disk files. (Model I)
DB to LII Converter, The	Mumford	1/81:200		Convert Disk Basic to Level II. (Model I)

Title	Author	Issue/Page	Debug	Annotation
Death Wish, The	Smith	12/81:262		Program that generates machine-language data statements as it self-destructs. (Model I)
Dexterous Data Entry	Shuken	12/81:280		Menu-generating routine. (Model I)
Direct Statement in File	Fitchhorn	5/81:190		Retrieve programs from disk after the "direct statement in file" error. (Model I)
Dirprog	Egbert	8/81:270		Disk directory program. (Model I)
Disk Auto-Menu, The	Hewin	6/81:178		Run and list programs from the menu. (Model I)
Disk Index	Schilling	8/81:268		Keep track of all your directory listings. (Model I)
Dollars and Sense	Andrews	5/81:216		Generate formatted input commands in business software. (Model I)
Driven Printer, A	Mueller	11/81:164		Assembly-language driver for the Centronics interface. (Model I)
El Cheapo Packer	Browne	3/81:168		Routine to pack two lines together or break one line in two. (Model I)
Emterm	Mills	10/81:156		Terminal program to use with ESI Lynx modem. (Model I)
Encoder, The	Cain	5/81:234		Convert hex into blocks of Basic that POKE hex back into proper memory locations. (Model I)
Enhance Your Level II Basic	Spencer	7/81:202		Assembly-language listing to add functions to Level II. (Model I)
Everyman's Mod II WP	Kilroy	7/81:226	10/81:22	A Basic word processor. (Model II)
EZ Wider	Chepko	3/81:247		Set printer character width and line length. (Model I)
Fast Edit	Illk	2/81:218		Juggle utilities to save time in reloading them. (Model I)
Find It Fast!	Yelvington	3/81:180		String-search for Basic programs. (Model I)
Flexible Scroller, The	Myers	5/81:204		Get segmented screen scrolls. (Model I)
Floppy Fixer	Beebe	12/81:326		Look anywhere on your disk and modify its contents. (Model I)
Full Error	Keairns	10/81:340		Get full error messages instead of abbreviations. (Model I)
Get High on Histograms	Lovy	1/81:211		Bar-graph generator. (Model I)
Head Bright	Hesse	9/81:327		Head-cleaning program. (Model I)
Hex Converter	Malone	5/81:244		Take hex off number base conversions. (Model I)
High-speed Data Tapes	Glosser	7/81:280	11/81:16	A quick string-array tape subroutine. (Model I)
II the Dump	Faber	12/81:236		Machine-language screen-dump routine. (Model II)
KBEEPFIX Revisited	Whitehead	3/81:270	6/81:16	Modify KBEEPFIX (2/80:14) for disk-based systems. (Model I)
KILDOS Is Here	Soltysik	6/81:265		Eliminate DOS from the disk to free space. (Model I and NEWDOS)
Know-It-All	Sehmer	5/81:288		List variables and lines in which they are used. (Model I)
Letter Counter	Atkins	3/81:192		Analyze text by tallying frequency of letters. (Model I)
Line Loss	Borrmann	5/81:283		Save the line lost to answering prompts. (Model I)
Listprog	Riffel	10/81:374		Improve your program listing's appearance. (Model I)
Logger, The	Kelly	12/81:374		Keep track of your programs. (Model I)
LPRINT Formatter II	Tzinberg	2/81:214		Print professional-looking listings. (Model I/III)
Memory Expander, The	Kenealy	9/81:174		Space saver that deletes spaces and remarks. (Model I)
Memory Window	Merkey	10/81:364		Machine-code subroutine to dump memory onto the screen. (Model I)
Merge for Level II	Dalesandry	11/81:284		Put a Merge utility on Level II cassette Basic. (Model I)
Mergers	Lloyd	12/81:238		Run RS's Mail List program in 48K. (Model I)
Modifying EDTASM +	Rhode	10/81:344		Store source and object code on disk. (Model I)
Never Ready	Balewski	7/81:199		Routine to change the Ready prompt. (Model I)
Newbug	Marks	11/81:368		Fifteen new commands for T-Bug. (Model I)
No Forwarding Address	Thiel	3/81:160		Protect your programs from tampering. (Model I)
Now You See It	Borrmann	2/81:236		Save and restore screen contents in 24 bytes or less. (Model I)
On Error. . .Fix	Richardson	9/81:250		Routine that detects and repairs errors. (Model I)

Title	Author	Issue/Page	Debug	Annotation
One-wheel Drive	Keen/Pezzuto	5/81:256		Copy utility for one disk drive. (Model I)
Paper Tiger Screenprinter	Lewart	2/81:138		Get screendumps of graphic display with the IDS 440 printer. (Model I)
Patching Across	Koch	9/81:282	12/81:26	Use Model I Scripsit on the Model III.
Pauper's Processor	Osburn	3/81:232		Simple Basic word processor. (Model I)
Purge	Ligori	6/81:219		Purge utility for disk. (Model I)
Ready—For the Model II?	Barbarello	12/81:260		Make your programs novice-proof. (Model II)
Rename	Busch	9/81:317		Get a rename utility on NEWDOS. (Model I)
Sans Disks	Hunter	4/81:186		Data-base management on cassette with fast loading and saving techniques. (Model I)
Sargon Saver, The	Quindry	5/81:272		Save Sargon in mid-game and play later. (Model I)
Sargon Saver, Pt. II	Quindry	12/81:348	4/82:26	A better way to save Sargon to tape. (Model I)
Scrip Patch	Allred	3/81:222		Go from Scripsit to DOS without losing text. (Model I)
Scripsit—Sans Serif	Cochrell	11/81:172		Use your printer's special features from Scripsit. (Model I)
Sentry, The	Rastin	12/81:242		Time-saving shift-entry utility. (Model I)
Serial Printing/ EDTASM+	Cohen	9/81:276		Driver to get serial printouts using EDTASM+. (Model I)
Shift Lock	Hambel	5/81:260		Assembly-language subroutine for shift reversal. (Model I)
Simplezap	Safford	11/81:360		Inspect your disk sectors. (Model I)
Single-drive File Copy	Gorsky	3/81:191		Copy utility for TRSDOS and one drive. (Model I)
Soft Tach	North	6/81:232		Display the rotational speed of your disk drives. (Model I)

1982

BUSINESS

Title	Author	Issue/Page	Debug	Annotation
Autotrak	Lloyd	10/82:203		Keep track of maintenance records for a fleet of vehicles. (Model I/III)
CC—Color Computer or Chamber of Commerce	Adcock	3/82:210		Using the Color Computer as a business machine.
Discriminating Pallets	Eaton	1/82:98		Plot prices for increasing material costs. (Model I)
Financial Wizard	Perelman	5/82:326		Figure compound interest, annuity, and amortization. (Model II)
High Finance	Byrne	10/82:324		Program to use with all-saver savings accounts. (Model I/III)
Industry Saver	Leichtman	10/82:156		How a TRS-80 improved production-line efficiency.
Invoice	Hackman	10/82:342		Organize your billing procedures. (Model I/III)
One Twelfth of a Misery	Krapf	4/82:298		Loan-amortization program. (Model I)
Order Form	Wright	9/82:270		Produce custom-designed forms. (Model I/III)
Phonfind	Eldridge	6/82:358		See Home/Hobby.
Pocket Portfolio	Dethlefsen	1/82:94		Investment-analysis program. (PC)
Portal-to-Portal Report	Keneipp	6/82:300		Keep track of air miles traveled. (Model I)
Production Learning Curve	Jeffrey	10/82:316		Program to track production costs per unit. (Model III)
Profit Prognosticator	Nottingham	2/82:268		Estimate business profits. (Model III)
Questions, Questions, Questions	Rutledge	1/82:102		Tabulate and average survey results for a business. (Model I/III)
QuickCalc	Leafstand	10/82:114		VisiCalc clone. (Model I/III)
Rule of 300	Zimmerman/Conrad	1/82:116		Figure interest payments on loans. (Model I/III)
Spanning of Mod II Disks	Keen/Dischert	1/82:110		See Programming Techniques.
Survey	Wells	6/82:248		Telemarketing program. (Model I/III)

Title	Author	Issue/Page	Debug	Annotation
Tax Relief for the Rich	Perelman	3/82:176		Figure taxes after the Economic Recovery Act. (Model II)
Trade Wins, The	Finkelstein	2/82:238		Stock Market simulator. (Model I/III)

EDUCATION

Title	Author	Issue/Page	Debug	Annotation
80 in the Apple, An	Radin	2/82:68		About a teacher in NYC using 80s in the classroom.
Anything Jodi Can Do...	Tallman	2/82:60		Code game written by 9-year old and comments from her dad on teaching kids to program. (Model I)
BIPED	Leichtman	4/82:112		Story on a program using micros to train handicapped people.
CIE—Computers in Education	Radin	6/82:254		How a school district uses computers.
Classroom Crystal Ball	Keough	2/82:94		The future of computers and education, and how programmers should view educational software.
Computer Etch-a-sketch	Mustico	2/82:70		See Programming Techniques.
Do Not Pass	Wood	6/82:352		Teach road signs in driver-education class. (CC)
Dynamic Item Scheduling	Wyckoff	5/82:316		Write educational programs that interest and teach students. (CC)
Earth to Class, Listen Up	Fish	2/82:120		Classroom networking. (Model I)
Elementary, My Dear Primate	Vanderburgh	2/82:256		Use randomly generated text to study language synthesis. (Model I)
Extra-terrestrial	Wells	2/82:112		Program to teach the solar system. (Model I)
Fret No More!	Louis	5/82:164		Learn how to play the guitar with your 80. (Model I)
Future in Miniature, The	Mello	2/82:53		The micro in the classroom.
Grade Book	Pugsley	3/82:184		Keep students' grades on the computer. (Model I)
King Komputer	Allison	4/82:302		Program to teach kids how to read a ruler. (Model I/III)
Learning the Elements	Wood	2/82:116		Program to teach chemical elements. (Model I/III)
Making More Possible	Leichtman	2/82:128		How micros help the handicapped.
Moby Dick Touch Typing Tutor	Brown	9/82:139		Learn touch typing. (Model I/III)
OJT	Tymon	6/82:326		Program to build your own educational programs from. (Model I)
Put Them to the Test	Davis	2/82:104		Program that generates quizzes. (Model I/III)
Roll Call	Henderson	2/82:100		Using computers to keep track of student records.
RS-80Tay, Aysay Hatway?	Stratton	2/82:142		Program to teach a foreign language. (Model I)
Ten-key Tutor, The	Knoyle	2/82:192		Program to teach typing on the ten-key pad. (Model I)
Time to Make 'em Sweat	Hawkes	5/82:332		Generate multiple-choice, true-false, short-answer, or completion tests. (Model I)
To Comma, or Not to Comma	Perron	2/82:82		Punctuation-teaching program. (Model I)
Vocabulary Test	Kalkstein	4/82:294		Match a word with a definition. (Model I)
You Light Up My Life	Wood	3/82:330		Teach the physics of light. (CC)

GAMES

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Acrostic Generator	Falk	8/82:240		Create puzzles. (Model I/III)
Alien Attack	Perry	8/82:282		Arcade game, with joysticks. (CC)
Amazing Cardoni II	O'Connor	8/82:302		Card trick. (Model I/III)
Assignment 46	Albino	8/82:66		Adventure game. (Model I)
Birthday Party	Zuckerbraun	10/82:262		Games for kids' birthday parties. (Model I)
Casino Slot Machine	Balewski	8/82:222		Simulate a Las Vegas slot machine. (Model I)
Color Breakaway	Grossbach	8/82:268		One-on-one hockey simulation. (CC)
Color Maze	White	8/82:188		Maze game. (CC)
Color Reversi	Ledger	3/82:90		Reversi on the Color Computer.
Computer Repeat	Fontenot	4/82:222	9/82:32	Memory game. (Model I)
Conquest of Memory Alpha	Myers	8/82:254		Colonize the universe. (Model I/III)
Cram	Brothers	8/82:234		Draw the line without going back or into it. (Model I/III)
Cube-80	Washington	8/82:106	10/82:30	Rubik's Cube on the computer. (Model I/III)
Enter the Dragon	Hadlock	8/82:248		Kung-fu adventure. (Model I/III)
Flip-a-piece	Cominio	3/82:252		Othello-like game. (Model III)
Fortran Puzzler	Yehle	6/82:227		Figure out the secret code. (Model I)
Four in One Plus Another	Becker	8/82:202	10/82:31	Four games and a utility for the CC.
Game of Kalah, The	Victor	8/82:132		Similar to Reversi. (Model I)
Game of Sim, The	Radin	9/82:276		Similar to tic-tac-toe. (Model I/III)
Intellectual Somnambulism	Keller	8/82:296		Arcade game. (Model I)
Kings and Catapults	Adams	2/82:232	9/82:32	Two fuedal kings battle it out. (Model I/III)
Loco Motion	Ridgway	8/82:286		One-player to align objects into a pattern. (Model I)
Martian Missile Attack	Gillen	1/82:265		Save cities, destroy Martians. (Model I)
Master Muses, The	Heath	8/82:186		See Tutorials.
Micro Puzzle Box	Moews	1/82:302	3/82:28	Put the numbers in the right order in a grid. (Model I)
Naval Wars	Byrnes	8/82:114		Battleship with two computers. (Model I/III)
Outdoor Computer Games?	Adams	8/82:80		Play spy vs. spy outdoors with your computer. (Model I/III)
Pitty Pat	Barnes	8/82:270		Variation of draw poker. (Model I/III)
Save All Humans	Boothe	3/82:154		Save people from flying saucers. (Model I/III)
Save Our Ship	Hawkins	8/82:174		Star Trek adventure. (Model I/III)
Ski Slalom	Commander	8/82:112		Hit the slopes on the 80. (Model I/III)
Space Chase	Gillen	5/82:292	10/82:30	Arcade space game. (Model I)
Space Duel	Edick	8/82:260		Arcade game. (Model I/III, CC)
Square Game	Kanach	8/82:264		Merlin-like game. (CC)
Stanley	Black	8/82:300		Out stare your VDT. (Model I)
Sub Destroy, Model II Style	Steele	8/82:212		Arcade game for the Model II.
Subchaser!	Steiner	3/82:106		Avoid depth charges and learn graphics techniques. (CC)
Subs 'n Choppers	Gillen	8/82:216		Arcade game, subs vs. choppers. (Model I)
Tee for Six	Bevington	8/82:156		Tee-jumping board games for the computer. (Model I/III)
Termites	Weindorf	8/82:274		Eat through the wood, but remember where the knots are. (Model I/III)
Thru the Asteroids	Fugate	8/82:280		Find your way through the asteroid belt and land safely. (Model I/III)
Tumblin' Dice	Bobo	2/82:166		Based on Boggle. (Model I)
You Light Up My Grid	McGlumphy	6/82:330		Light all the squares but the center one. (Model I)

GENERAL

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Basic Misinterpreter	Busch	4/82:86		Humor—change Basic keywords to misspelled words. (Model I)
Bob Rosen—A Colorful Story	Leichtman	6/82:174		Personality piece on Bob Rosen.

Title	Author	Issue/Page	Debug	Annotation
Broadening the TRS-80 Horizons	Busch	3/82:298		Thoughts on expanding your system.
Celluloid CPUs	Nadeau	5/82:98		Micros in the movies.
CompuServe, MicroNet and the TRS-80	Maguire	1/82:74		TRS-80 users and CompuServe.
Computer Creationists	Rose	5/82:80		How micros are used in sound studios.
Confessions of a Microholic	Keller	5/82:320		One user's life after buying a TRS-80.
Dateline: Sri Lanka	Mello	4/82:128		Personality piece on David Busch.
Divine Dementia	Nadeau	5/82:176		Personality piece on Dennis Kitsz.
Dream Team	Wangsness	2/82:196		Put together an ideal team with this program. (Model I)
Gabby the Space Cowbum	Ramella	5/82:322		This space cowboy won't shut up. (Model I)
Graphics King, The	Frann	8/82:120		Personality piece on Leo Christopherson.
Jake's Wampeters	Mello	3/82:162		Profile on Jake Commander.
Kryha Cipher Machine	Deavours	5/82:272		Use Tiny Pascal in cryptology. (Model I)
Little Byte Music, A	Levine	5/82:128		History of digitally synthesized music.
Night on the Town, A	Gunn	4/82:124		Humor—take your computer out on a date.
Northern Lights	Latamore	5/82:104		How a sculptor uses a micro to plan his designs.
Ohio Electronic News Experiment	Chidsey	6/82:100		Progress report on the Tiffin, OH, newspaper's electronic news experiment.
Out of Thin Error	Adcock	5/82:258		Humor—what error messages "really" mean.
Philly Phiasco	Gunn	5/82:140		Commentarty on the Philadelphia Computer in the Arts Symposium.
Radio Shack vs. IBM	Van Ghent	3/82:168		Model II compared to the IBM 5150.
ROM Bibliography	Secord	4/82:76		Annotated list of literature on the ROM. (Model I/III)
Shopping with Uncle Sam	Stolker	10/82:88		A look at the Federal DP Expo in Washington, DC.
Terminal Case, A	Latamore	1/81:66		A look at the Canadian videotext system, Telidon.
TRON: Man in the Computer	Mello	8/82:124		Piece on the movie, Tron.
Vexed by the Void	Resnick	4/82:262		An alien's view of TRS-80 users—humor.
Videotext for the Masses	Nadeau	1/82:60		What videotext means to the home computerist.
When the Postman Doesn't Ring	Averill	3/82:308		Thoughts on mail order.

GRAPHICS

Title	Author	Issue/Page	Debug	Annotation
3-D Graphics	Fogelin	3/82:138		Get 3-D geometric shapes on the screen. (Model I/III)
Alpha-graphics	Basch	1/82:190		See Programming Techniques.
Back to the Drawing Board	Rappaport	3/82:120		See Utilities.
CC on Parade, Pt. I	Barden	10/82:82		Using graphics on the CC.
Color from a Model II?	Ward/Deninger	3/82:240		Use the DMP-3 digital plotter to get color graphics. (Model II)
Colorful Computer, Pt. I	Miller	8/82:94		Twenty-one graphics programs. (CC)
Colorful Computer, Pt. II	Miller	9/82:152		See Part I.
Colorful Computer, Pt. III	Miller	10/82:254		See Part I.
Computer Etch-a-sketch	Mustico	2/82:70		See Programming Techniques.
Conversion	Osborne	9/82:238		Bob Boothe's graphics on the CC.
Different Perspective	Nielsen	5/82:242		Use shadows to gain perspective in graphics. (Model I)
Editor's Choice, The	Gesamte/Commander	3/82:78		Graphic patterns. (CC)
Is a Rose in Color/ a Rose?	Green	3/82:142	6/82:26	Earlier *80 Micro* graphics programs converted to the CC.
Joystick Paintbrush	Sprouse	9/82:230		Draw on the screen with joysticks. (CC)
Mirror Imagining	Boothe	3/82:112		Plot spheres, paraboloids, and other high-resolution shapes for the Epson. (Model I)
Paper Graphics	Rosenberg	3/82:270		See Utilities.
Pictures at a Mod II Exhibition	Baker	3/82:280	6/82:26	Bob Boothe's techniques on a Model II.

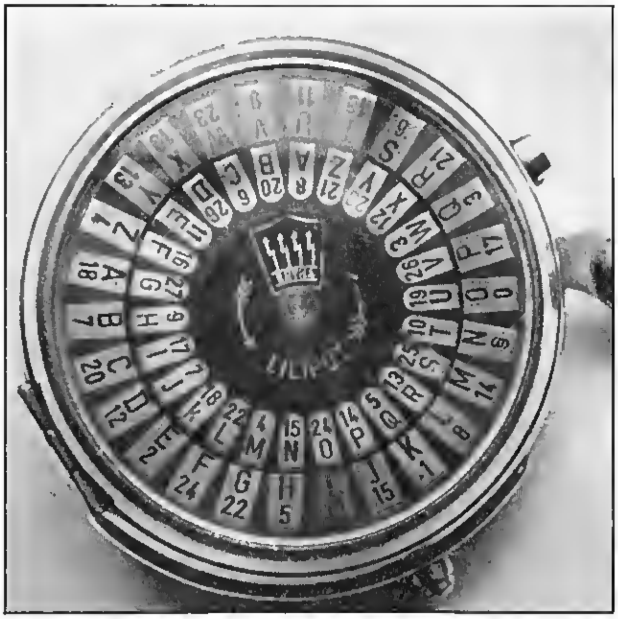
Polar Generator	Webb	2/82:204		See Programming Techniques.
Random Picture Generator	Swarts	2/82:194		Let the computer be the artist. (Model I)
Shady Characters	Ramella	3/82:258	6/82:26	Get silhouette printouts. (Model I)
Smooth Graphics	Goodman	3/82:304		See Programming Techniques.
Spiromania, Pt. I	Commander	5/82:88		Draw spirographs on the Color Computer.
Spiromania, Pt. II	Commander	6/82:106		See Part I.
Sublime Simulations	Keough	4/82:258		How computer simulation/modeling works.
Super Banner	Balewski	5/82:282		Print out banners on a Centronics 737. (Model I)
When Black Is White	Tache	3/82:294		Reverse graphic printouts. (Model I/III)

HARDWARE

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Bare-bones Communicator	Hart	6/82:128		Get a communications system without the expansion interface or RS-232 adapter, with software. (Model I)
Battery Back-up	Batie	2/82:126		Build a battery back-up unit.
Building an M-80	Hawkes/Reese	3/82:172		Build a single-board microprocessor with the Z80.
Cheaper Upgrade	Tucker	9/82:186		Install the Extended Color Basic ROM yourself. (CC)
Color Computer Upgrade	Murphy	3/82:102		Go from 4K to 16K. (CC)
Color from a Model II?	Ward/Deninger	3/82:240		See Graphics.
Computerized Engraving	Joffe	5/82:318		Drive a pantograph with your 80. (Model III)
Cybernetics and Jelly Beans	Davids	10/82:190		Build a robot to detect jelly beans. (Model I)
Digital Doodles	Sehmer	1/82:244	9/82:30	Build a graphics plotter. (Model I)
Do-it-yourself Disks	Schaefer	1/82:172		Install your own disk drives in a Model III.
Double Your Density	Domuret	1/82:294		Adding double density to your system section. (Model I)
Four Into One Will Go	Hawkes/Reese	2/82:226		Put a 4K program in 1K by bank. (Model I)
Handy Dandy Tandy Table	Langston	4/82:328		Build a desk for your computer.
Hardware Hacker, Pt. I	Van Praag	10/82:216		Connect a Centronics-compatible printer to the Model I.
Hydra-disk	Robins	3/82:206	6/82:26	Add dual-headed drives. (Model I)
Joy of Interfacing, The	Batie	3/82:242		Joystick interface. (Model I)
Juicing Pin 18	Gorodetzer	9/82:288		Use pin 18 on the MX-80 printer connector with the buffered cable. (Model I)
Look into Disk Drives, A	staff	1/82:179		Buyer's guide to disk drives.
Modem Auto-answer	Westbrook	6/82:229		Build an auto-answer device.
Multi-programming on a Micro	Genovese	1/82:278		See Programming Techniques.
Networking on a Shoestring	Meinke	2/82:184		Build your own networking system. (Model I)
New Generation of Characters	Park	4/82:220		Install the AXX3027 character generator chip. (Model I)
Programmable Baud Rate	Cottle	5/82:306		Device for the LNW interface to get a programmable baud rate. (Model I)
Programmable Sound Generator	Robins	5/82:106		Get up to nine simultaneous voices for about $125. (Model I)
RAM Wars	McClenahan	3/82:156		Get 64K RAM on the CC.
Singer Printer Interface	Mailhot	6/82:374		Interface the Singer printer.
Sixteen-channel A/D Board	Haan	6/82:310		Interface eight joysticks. (Model I)
Smarten Up, Color Computer	Esposito/Thiel	3/82:126		Add memory to the CC.
Straight Shooter, The	Quindry	1/82:318	10/82:30	Build an inexpensive light pen. (Model I)
Telephone Dialer	Hickey	6/82:160		Device to let your computer dial phone numbers. (Model I)
Those CLOAD Blues	Hartjes	1/82:288		Build a data compensator and audio amplifier. (Model I)
Video Snow Shovel	Smith	3/82:290	9/82:30	Get rid of those black streaks against white characters. (Model I)
Where There's a Will...	McClenahan	3/82:84		Device to let the Color Computer print while on CompuServe.

HOME/HOBBY

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Bringing the Supermarket to Its Knees	Kraynak	4/82:270		Grocery-list program. (Model I)
Computer Racing Form	Bobo	5/82:252		Handicap horse races on your micro. (Model I/III)
Fore! Scorekeeper	Wells	5/82:298		Keep track of your golf scores. (Model I/III)
Gentle Reminder, A	Rosen	6/82:348		Schedule your work projects. (Model I/III)
Home Budgeteer—Reprise	Pablo	5/82:238		Keep track of household expenses. (Model I/III)
If This Is Tuesday, It Must Be. . .	Atkins	1/82:308	5/82:26	Date program. (PC)
Income Tax Estimator	Stark	10/82:168		Figure your income tax on the CC.
Model III Biorhythms	Anderson	5/82:280		Find out what your good and bad days will be. (Model II)
No More 90-pound Weakling	Stevens	1/82:174		Excercise log program. (PC)
Phonfind	Eldridge	6/82:358		Phone-log program. (Model I/III)
To Catch a Thief	Kuhn	4/82:274		Home-inventory program. (Model I/III)
Trick or TRS-80	Keller	10/82:104		Have a jack-o-lantern on your screen greet trick-or-treaters. (Model I/III)
Two Strokes a Side	Avery	5/82:264		Figure your golf handicap. (Model I/III)
Ultimate Parts Manager	McCalley	1/82:140		Keep inventory of Model T parts. (Model I)
You're in the Money	Montgomery	10/82:140		Project the impact of personal savings and borrowing plans. (Model III)



PROGRAMMING TECHNIQUES

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Add That Professional Touch	Steelhammer	1/82:258		Make DBMs easier to use. (Model I/III)
Alpha-graphics	Basch	1/82:190		Technique to build graphic strings. (Model I/III)
Basic Word Processing	Cutrona	9/82:140		See Utilities.
CC on Parade, Pt. I	Barden	10/82:82		See Graphics.
Colorful Computer, Pt. I	Miller	8/82:94		See Graphics.
Colorful Computer, Pt. II	Miller	9/82:152		See Part I, Graphics.
Colorful Computer, Pt. III	Miller	10/82:254		See Part I, Graphics.
Computer Etch-a-sketch	Mustico	2/82:70		Add graphic routines to educational programs. (Model I)
Conversion	Osborne	9/82:238		See Graphics.
Different Perspective	Nielsen	5/82:242		See Graphics.
Dizzy Decimals	Shore	3/82:326		Eliminate round-off errors. (Model I/III)
Do-it-yourself DB, Pt. I	Townsend	6/82:176		How to write your own data base.

Title	Author	Issue/Page	Debug	Annotation
Do-it-yourself DB, Pt. II	Townsend	9/82:242		See Part I.
Do-it-yourself DB, Pt. III	Townsend	10/82:332		See Part I.
Editor's Choice, The	Gesamte/Commander	3/82:78		See Graphics.
Flasher, The	Mickey	1/82:276		Flash prompts. (Model I)
I Have a Secret	Demberger	10/82:296		See Utilities.
I Program, Therefore ISAM	Adcock	5/82:302		Use the indexed sequential addressing mode.
Mirror Imagining	Boothe	3/82:112		See Graphics.
Model I Do Files	Jackisch/Knapp	4/82:254		Ease program operation by using Do files. (Model I)
Multi-programming on a Micro	Genovese	1/82:278		Execute several programs simultaneously. (Model I)
Ordered Chaos	Webb	1/82:310		Using randomness in programs. (Model I/III)
PCLEAR 0	Heusinkveld	9/82:282		Make high-resolution graphics use high memory. (CC)
PEEK of Its Career, The	Wilson	6/82:308		Simulate PEEK and POKE on the Model II.
Performance Analysis	Ballard	2/82:240		Detail the execution path of your programs. (Model I/III)
Polar Generator	Webb	2/82:204		Generate a map of the Arctic on the screen. (Model I)
Priming the Data Base	Ring	3/82:152		Use prime numbers for labels in DBMs for fast sequential searches. (Model I/III)
Printer Color Art	Kalinowski	9/82:168		Get color printouts on your MX-80 using colored ribbons. (Model I/III)
Programmers for Hire	Gillig	1/82:274		Techniques of a professional programmer. (Model I/III)
Programming Pitch, Pt. I	Davis	5/82:142		How to program pitch in music-generation programs. (Model I)
Programming Pitch, Pt. II	Davis	6/82:362		See Part I.
Programming Pitch, Pt. III	Davis	10/82:228		See Part I.
ROM Breakout	Sprott	6/82:350		Use the interpreter's built-in exits to make modifications. (Model I/III)
Sling Some Hash	Knecht	9/82:124		Use hashing to store and retrieve items in an unsorted list. (Model I/III)
Smooth Graphics	Goodman	3/82:304		Add precision to animation. (Model I/III)
Spanning of Mod II Disks	Keen/Dischert	1/82:110		Link files with multi-disk Model IIs.
Stepwise Refinement	Boasso	6/82:232		Excercise using psuedo-code.
STRING$ the Thing, The	Knight	6/82:298		Use STRING$ to pack strings. (Model I/III)
Subchaser!	Steiner	3/82:106		See Games.
Super Banner	Balewski	5/82:282		See Graphics.
To Err Is. . . Okay	Adcock	3/82:230		Use On Error GOTO to work out of programming problems.
Total Recall	Bender	4/82:332		Get direct access to anything in memory with Fortran. (Model I)
Using Print Using	Rende	1/82:290		Tips on using the Print Using command.
Worm Pills for Basic	Keen/Dischert	4/82:290		Find the most efficient way to use Basic keywords.

SCIENCE/MATH

Title	Author	Issue/Page	Debug	Annotation
Bemusing Triangle, The	Honess	2/82:210		Use the TRS-80 to implement the trapezoidal rule. (Model I/III)
Colorful Titrations	Wood	2/82:202		Isolate specific elements in chemical solutions. (CC)
DIGRAPH Digressions	Gorney	6/82:192		Use a directed graph to learn graph theory. (Model I/III)
Extra-terrestrial	Wells	2/82:112		See Education.
Learning the Elements	Wood	2/82:116		See Education.
Model III Biorhythms	Anderson	5/82:280		See Home/Hobby.
Propagation Prediction	Chipman	6/82:272		Predict high-frequency wave propagation. (Model I)
Two-way ANOVA	McGarvey	3/82:234		Get two-way analysis of variance. (Model I)
You Light Up My Life	Wood	3/82:330		See Education.

TUTORIALS

Title	Author	Issue/Page	Debug	Annotation
BBS Primer	Wright	6/82:284		What you need to hook onto a bulletin board.
Bit Smitten, Pt. II	Chidsey	9/82:268		Formerly "For the Novice;" heat build-up, turn-on and turn-off shock, and more discussed.
Bit Smitten, Pt. III	Chidsey	10/82:96		What a baud is, screen-dump routines, and more.
Breaker 19	Busch	6/82:96		How to use CompuServe's CB simulator.
Damping Cassette Output	Sinclair	5/82:324		What to do when CLOAD problems are really CSAVE problems.
Data Communications—TRS-80 Style	Derfler	6/82:82		How to set up your TRS-80 as a data-communications terminal.
Do-it-yourself DB, Pt. I	Townsend	6/82:176		See Programming Techniques.
Do-it-yourself DB, Pt. II	Townsend	9/82:242		See Part I, Programming Techniques.
Extended Color Basic	Miller	6/82:266		A look at Extended Color Basic. (CC)
For the Novice, Pt. I	Chidsey	6/82:148		What MEM SIZE?, high and low memory, high and low-level languages are all about.
Graftrax 80	McNamee	9/82:190		Use bit graphics on the MX-80. (Model I/III)
Inside Scripsit, Pt. I	Lindley	9/82:222		Learn how Scripsit works before you modify it. (Model I/III)
Inside Scripsit, Pt. II	Lindley	10.82:276		See Part I.
Learn a Little Pascal, Pt. II	Grothman	1/82:80		Program using RS Tiny Pascal.
Make Butterflies—Not Bugs	Commander	4/82:152		Tips on using utilities.
Master Muses, The	Heath	8/82:186		Author of Master Reversi talks about computer Othello.
Mod III Notes	Ratzlaff	2/82:200		Model II monitors and cassette I/O.
Model I, Meet Model III	Barlow/Brydges	1/82:316		Convert programs from Model I to Model III.
Model II Primer	Baker	5/82:260		Tips for the new Model II owner.
Ordered Chaos	Webb	1/82:310		See Programming Techniques.
Printing Perfection	Phillip	9/82:126		Control print density and size options of the MX-80 while in Scripsit.
Radio Shack Repairs	Kepner	10/82:101		What Radio Shack charges to repair specific things.
Stepwise Refinement	Boasso	6/82:232		See Programming Techniques.
Technological Destiny, Pt. I	Dillio	4/82:264		What a job in data processing requires.
Technological Destiny, Pt. II	Dillio	5/82:228		See Part I.
Technological Destiny, Pt. III	Dillio	6/82:186		See Part I.
Technological Destiny, Pt. IV	Dillio	9/82:260		See Part I.
Technological Destiny, Pt. V	Dillio	10/82:246		See Part I.
Using Print Using	Rende	1/82:290		See Programming Techniques.
Utilities Buyer's Guide	staff	4/82:135		Stats on various utilities for the TRS-80. (Model I/II/III, CC)
Welcome to Cobol	Keen/Dischert	1/82:92		Introductory look at Cobol.
Worm Pills for Basic	Keen/Dischert	4/82:290		See Programming Techniques.
Z80 Bit Tables	Robinson	3/82:260		Op-code secrets revealed. (Model I)

UTILITIES

Title	Author	Issue/Page	Debug	Annotation
16, 10, 2, or 8—Which Base Do We Appreciate?	Sarnow	6/82:334		Decimal to octal to hex conversion. (Model I/III)
Add CRC ASAP	Baker	1/82:204		Perform cyclical redundancy code checks for tape-based systems. (Model I)
Another Major Operation on Scripsit	Graves	1/82:230		Add serial printer capabilities to Scripsit. (Model I)
Array I/O	Neff	1/82:200	5/82:26	Machine-language routine for fast Basic data transfer. (Model I)

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
As the Screen Scrolls	Keller	2/82:264	6/82:26	Protect some of your screen from scrolling. (Model I)
Back to the Drawing Board	Rappaport	3/82:120		Design graphic displays on the screen and let the computer produce the code for it. (Model I)
BAM!	Byrkit	4/82:167		Change one-letter commands in your monitor program to a full word. (Model I)
Base 2 Screenprinter	Kent	3/82:214		Get screen dumps on the Base 2 printer. (Model I)
Basic Communication	McGarvey	6/82:324		Terminal program in Basic. (Model I)
Basic Compiler in Basic	Bertsekas	10/82:122		(Model I)
Basic Translator	Miller	6/82:194		Translate different Basic dialects for communication with other computers.
Basic Word Processing	Cutrona	9/82:140		Basic word processor. (CC)
Can You Get Me a Date?	Phillipp	2/82:220		Routine to let you enter a date in any format you like. (Model I)
Colormon	Cook	3/82:212		Monitor program. (CC)
Command Interpreter	Alford	4/82:244		Directly access machine-language routines. (Model I)
CP80	Cameron	4/82:306		IBM-like monitor program. (Model I)
Datagen	Heusinkveld	6/82:346		Convert machine code to Basic statements. (CC)
Direct Access	Fink	4/82:214		Access Disk I/O routines from Basic using string variables as input buffers. (Model I)
Does Format Get Your Backup?	Hart	1/82:217		Use glitched disks. (Model I)
Error Code Expanded	Alford	2/82:260		Get full error messages. (Model I)
Expand It—Burn It In	Alexander	6/82:344		Test program for memory upgrade. (Model III)
Fill in the Blanks	Schuldenfrei	1/82:224		Pack records before sending them to tape. (Model I/III)
Graphics for Profile	Wood	3/82:284		Get graphics with Model II Profile.
Half Duplexer	McGarvey	5/82:172		Get screen echo for typed communications. (Model I)
Hoodwinking TRSDOS	Anderson	6/82:296		CP/M printer driver for Model II TRSDOS.
Horizontal Scrolling	Foley	6/82:318		Scroll back and forth and up and down. (Model I)
I Have a Secret	Demberger	10/82:296		Data encryption-decryption program. (Model I/III)
JKL Minus Blanks	Straw	9/82:290		Modify the NEWDOS+ JKL so it doesn't print blank lines. (Model I)
Loc-Editor	O'Connor	4/82:206		A spelling checker for program listings. (Model I)
Lost and Found	Athanasiou	6/82:288		Disk-directory program. (Model III)
Lost in Basic	Paxton	1/82:304	5/82:26	Find variables in Basic programs. (Model I)
Lots of Little Letters to Litter Your Listings	Olsen	2/82:262		Convert upper to lowercase. (Model I)
LP VII Patch for the CC	Degler	10/82:304		Eight-bit printer driver. (CC)
Memory Size?	Jackson	4/82:226		Store machine-language routines as string variables in Basic programs. (Model I)
Micro Melodies	Gibbs	5/82:234		Music-generation program. (Model I)
Model II Disassembler	Faber	4/82:182		A disassembler for the Model II.
Model II Terminal Driver	Korenthal	4/82:176		Terminal driver for use with Lifeboat's CP/M. (Model II)
Model III Master Directory	Muehlig	4/82:250		A master-directory program in 32K. (Model III)
Neatlist	Ewart	1/82:196		Produce neat program listings. (Model I)
One-drive Bulletin Board	Hodgson	3/82:314		Set up a CBBS with one disk drive. (Model I)
Paper Graphics	Rosenberg	3/82:270		Get graphics printouts on an LP IV. (Model I)
Play a Trick on Profile	Scott	6/82:306		Directly access profile files. (Model I)
Print That Index	Everett	6/82:184		Print out an address file. (Model I/III)
Print Whiz	Parker	2/82:250		Routine to use Electric Pencil and the RS LP IV to their full extent. (Model I)
Program Begat, Son of Program Begat	Christensen/Sater	6/82:320		Self-reproducing program. (Model I)
Rapid System Loader	Hedinger	4/82:188		Software tape-load speedup. (Model I)
Reading, Writing, and AL	Morgan	3/82:318		Routines for disk I/O. (Model I/III)
Recover	Gobel	9/82:256		Recover from unwanted exits from Scripsit. (Model I)
Screen Veil	Keller	9/82:286		Temporarily hide your screen's contents. (Model I/III)
Screenplay	Merkey	9/82:204		Print out anything in memory. (Model I)

Title	*Author*	*Issue/Page*	*Debug*	*Annotation*
Secrets	Mills	3/82:216		Hide sensitive data and protect your software. (Model I)
Snapshot	Rice	2/82:272		See how the values of your variables change during program execution. (Model I)
Systemized Basic	Browne	4/82:234		Save Basic programs in System format. (Model I)
Tab Extender	Hedinger	2/82:248		Put 127 more spaces on your tab limit. (Model I)
Tape Spate	Bowman	1/82:240		Shorten time for cassette data saves and recoveries. (Model I)
TRSDOS 2.0 Fix	Mornini	3/82:332		Keep from exiting to DOS. (Model II)
Two Transfers Please	Hollins	2/82:188		Put Eliza and Micromusic on disk. (Model I)
Using Test1A	Moultrie	4/82:240		Use TRSDOS's memory test with other DOSes. (Model I)
Varispeed	Evans	3/82:336		Control the speed of your Basic interpreter. (Model I)
Zubroutines	Ashley	4/82:209		Fifty short subroutines to aid your programming. (Model I)
Lowercase Done Right	Burgan	9/81:298		Build a better LC mod. (Model I)
Model I½, The	Fortna	4/81:218		Install your Stringy Floppy in the TRS-80 as one system. (Model I)
Model I as a Dumb Terminal	Tallman	4/81:276		Level I or II communications system. (Model I)
One Man's Robot	McAllister	9/81:114		Build a light-sensitive robot.
Onomatoeighty	Mein	1/81:190		Interface the General Instruments AY-3-8910 programmable sound generator. (Model I)
Plug-compatible Processor	Vonk	6/81:137		Z80A black box to make your Model I run at 4 MHz.
Polyphonic Sound Synthesizer	Brocaw	6/81:166	3/82:18	Complex sound generation using the General Instruments AY-3-8910 Programmable Sound Generator. (Model I)
Pulse Jockey, The	Suter	5/81:236		Interface via serial I/O through the cassette port. (Model I)
Real World Interface, Pt. I	Rand	10/81:202		Build an interface device. (Model I)
Real World Interface, Pt. II	Rand	11/81:208		See Part I.
Real World Interface, Pt. III	Rand	12/81:192		See Part I.
Recipe for Hardcopy	Keith	7/81:262		Interface a Teletype Model 33 with a Multi-80 and software. (Model I)
Red Letter Day/Lowercase	Ferber	4/81:248		Installing commercial lowercase printer and video mods and patches. (Model I)
Robotics—The Micro Connection	Brown	9/81:104		How robotics are edging into the micro field.
ROM Roll-over	Kelch	11/81:362	4/82:24	Mod to roll 16K of RAM over the ROM operating system. (Model I)
Second Sourcing	Tinis	11/81:348		Program to screen op-amps. (Model I)
See No Evil	Macri/Gregory	3/81:214		Monitor status of the CPU when the video isn't hooked up. (Model I)
Switched-on CLOAD	Bickel	3/81:252		Play tapes without unplugging wires from the keyboard. (Model I)
Tandy Acquires IBM!	Stanley	12/81:268		Interface the Selectric Model 1980 typewriter to the TRS-80. (Model I)
Tape Regenerator	Lewart	10/81:196		Back up even poor tape recordings on a second recorder with this device.
That Annoying Twitch	Smith	10/81:310		Device to eliminate video twitch. (Model I)
TRS-80 Bus Conductor	Fox	4/81:272		Build an expansion bus. (Model I)
TRS-80 Joystick Control	Barker	4/81:262		Interface Atari joysticks. (Model I)
Very Versatile Interface	Stanley	2/81:175		Build an interface board. (Model I)
Wandering 80, The	Hubert	8/81:298		Build a cabinet for your Model I.
Wave Shaper	Shreve	9/81:218		Build a device to square pulses during CLOADs.
Width Control	Barbarello	9/81:258		Build a control to set repeatable character widths on RS line printer.
Xerox 1740 and the Mod II	Nestor	6/81:92		Interface Xerox 1740 printer to Model II.

80 Reviews 1980

Books

Product	*Manufacturer*	*Issue/Page*
1001 Things to Do w/ Personal Computers	TAB Books	12/80:32
80 Programs for the TRS-80	Wayne Green Inc.	4/80:20
80-US	80 NW Publishing	2/80:38
An Intro. to Computer Music	John Wiley & Sons Inc.	12/80:32
Computer Games for Business and School	Winthrop Publishers Inc.	9/80:20
Freelance Software Publishing	Kern Publications	2/80:15
Guide to TRS-80 Information	F. E. Hubener	2/80:38
Inside Level II	Mumford Micro Systems	10/80:32
Introduction to TRS-80 Graphics	Dilithium Press	4/80:24
Introduction to T-Bug	Dilithium Press	7/80:16
Learning Level II	Compusoft Publishing	5/80:12
Micro Millenium, The	Viking Press	8/80:20
Most Popular Subroutines in Basic	TAB Books	11/80:38
Pascal-Intro to Logical Programming	Computer Science Press	5/13:13
Periodical Guide for Computerists	E. Berg	2/80:39
Problem Solving/Structured Programming	Addison-Wesley	5/80:13
Programming Techs for Level II Basic	Tandy/Radio Shack	11/80:38
Programmer's Book of Rules, The	Lifetime Learning Publications	5/80:14
Recreational Computing	Peoples Computer Co.	2/80:39
Running Wild	Osborne/McGraw Hill	4/80:24
Software Buyer's Guide	Wallace Electronics	2/80:39
SSI Micro Software Guide, The	SSI	2/80:38
Supermap	Fuller Software	7/80:16
TRS-80 Disassembled Handbook	Richcraft Engineering	6/80:18
TRS-80 Interfacing	Blacksburg Education Series	9/80:20
TRS-80 Monthly Newsletter	H&E Computronics	2/80:38
TRS-80 Technical Reference Manual	Tandy/Radio Shack	5/80:12

Hardware

Product	*Manufacturer*	*Issue/Page*
Acu-Data Tape Digitizer	Alphanetics Mfg.	6/80:14
Anadex Printer	Anadex	2/80:79
Beta-80	Meca	5/80:82
Centronics 730 Printer	Centronics Inc.	7/80:124
Comprint 912	Computer Printers International	11/80:40
DB-9500 Line Printer	Anadex Inc.	10/80:39
Exatron Stringy Floppy	Exatron	5/80:82
High-speed Modification Kit	Simutek	11/80:39
Mayday +S	Sun-Research Inc.	6/80:16
Microline-80	Okidata	10/80:39
Model 440 Paper Tiger	Integral Data Systems Inc.	10/80:36
Model 800 Printer	Base-2	9/80:24
Percom Disk Drives	Percom	2/80:66
Quick Printer	Centronics Data Computer Corp.	3/80:77
RS-232 Board	Tandy/Radio Shack	3/80:136
TC-8 Cassette System	JPC Products	6/80:14
Trendcom 100 Printer	Trendcom	8/80:18
TRS-80	Tandy/Radio Shack	8/80:184
TRS-80 Model II	Tandy/Radio Shack	7/80:20
TRS-80 Voice Synthesizer	Tandy/Radio Shack	9/80:154

Software

Product	*Manufacturer*	*Issue/Page*
Adventure	Softwin Assoc/Microsoft	6/80:18
Air Flight Simulation	Instant Software Inc.	4/80:24
Android Nim	80-NW Publishing Co.	2/80:60
Applications	Dilithium Tapes	4/80:136
AUTOK and QEDIT	Discovery Bay Software	2/80:58

Product	Manufacturer	Issue/Page
Basic 1P	Small Systems Software	5/80:14
Bootstrap	Practical Applications	7/80:17
Business Mail System	The Bottom Shelf	2/80:24
C Basic	FMG Corp.	4/80:130
Disk Directory	Mumford Microsystems	1/80:34
Disk Instruction Course	Tandy/Radio Shack	11/80:44
Disk Mod	Misosys	12/80:35
Doctor and Fetch	Omicron Software	1/80:48
EDAS 4.0	Galactic Software	8/80:20
Educational	Dilithium Tapes	4/80:136
ESP-1	Small Systems Software	2/80:59
Games	Dilithium Tapes	4/80:136
Generalized Subroutine Facility	Racet Computes	1/80:49
Gomoku	Discovery Bay Software	2/80:80
Graphics and Miscellaneous	Dilithium Tapes	4/80:136
IDM-IV Data-base Manager	Micro Architect	11/80:44
Individual Study Center	TYC Software	4/80:20
Inseq-80 and Insort-80	S & M Systems Inc.	10/80:33
Interlude—Ultimate Experience	Syntonic Software	9/80:21
ISAM	Johnson Associates	9/80:22
ISAR, Information Storage & Retrieval	The Alternate Source	12/80:36
Keepit Version 2.0	The Alternate Source	12/80:33

Product	Manufacturer	Issue/Page
Level I in Level II	Apparat Software	3/80:134
Level III	Microsoft	5/80:58
Line Renumbering	Software Association	1/80:48
Mail/File List	Galactic Software	2/80:24
Maillist	DAR Sales	10/80:36
Mathematics	Dilithium Tapes	4/80:136
Maxi-Disk and Shuffleboard	Parasitic Engineering	5/80:15
Micro Music	Tandy/Radio Shack	10/80:35
Micro-Opoly	Level IV Products Inc	10/80:35
MMS Forth	Miller Microcomputing Services	11/80:45
MON-2	Hubert Howe	2/80:59
Morse Code Transmit & Receive Program	Richcraft Engineering	7/80:100
Moving Signboard	Circle Enterprises	2/80:58
Music Composer/Editor	PFDC Software	1/80:49
Name and Address System	Small Business Systems Group	2/80:25
OBJREL	Hubert Howe	2/80:59
People's Pascal I and II	Computer Information Exchange	5/80:38
Planetary Lander	Instant Software Inc.	9/80:21
Poor Man's Text Editor	Don Coon	2/80:111
Pyramid	Tandy/Radio Shack	8/80:21
Radex-10	IJG Computer Services	7/80:18
Radio Shack Mailer	Tandy/Radio Shack	2/80:26
Remodel-Proload	Racet Computes	3/80:58

Product	Manufacturer	Issue/Page
Renumber	Tandy/Radio Shack	3/80:58
RSM-2 Monitor	Small Systems Software	4/80:70
Scripsit	Tandy/Radio Shack	5/80:56
Sort-II	Northeast Microwave	2/80:59
Special Delivery	Software Etc.	7/80:16
Step by Step. .	Program Design Inc.	2/80:59
T-Short	Web Associates	4/80:23
TRS-232 Formatter	Small Systems Software	10/80:32
Video Checkers	Compu-Quote	2/80:98
Win 21	Discovery Bay Software	2/80:97
Word-1	Micro Architect	6/80:92
Word-IV—Disk Word Processor	Micro Architect	11/80:44
Z80 ZAP/CMD	Org-Tex Industries	9/80:22

80 Reviews 1981

Books

Product	Manufacturer	Issue
57 Practical Programs & Games/Basic	Tandy/Radio Shack	4/81:31
A Safe House Electronically	Tandy/Radio Shack	2/81:20
Magazine of Advanced Applications	Alternate Source, The	1/81:47
Basic Fortran	Hayden Book Co.	6/81:36
Basic Scientific Subroutines	Byte Publications	9/81:24
Creative Kids Guide/Home Computers	Doubleday & Co.	4/81:31
Data File Programming In Basic	John Wiley & Sons	11/81:30
Heathkit Assembly Lang Course	Heath Corp.	9/81:146
Inside Basic Games	Sybex Inc.	9/81:22
Intro To Pascal, UCSD Pascal	Sybex Inc.	11/81:28
Microcomputing Techniques	Reston Publishing Co.	3/81:18
Mindstorms: Children, Computer	Basic Books Inc.	2/81:14
More TRS-80 Basic	John Wiley & Sons	12/81:82
Morse Code, Baudot and ASCII	Richcraft Engineering, Ltd.	11/81:34
Musical Applications of Micro.	Hayden Book Co.	9/81:22
Nature of Computation: Intro	Computer Science Press	10/81:36
New Uses For Home Comp/Stock Mkt	Thomas V. Lenz	11/81:30
Owning Your Home Computer: Guide	Everest House	6/81:34
Pascal	Matrix Publishing	4/81:31
The Small Computer In Small Business	The Stephen Green Press	11/81:31
Structured Basic and Beyond	Computer Science Press	11/81:28
The CP/M Handbook With MP/M	Sybex	3/81:18
TRS-80 Assembly Language	Prentice-Hall Inc.	10/81:36
TRS-80 Level II Basic, T Guide	Tandy/Radio Shack	1/81:47
UCSD Pascal Compiler	FMG Corp.	10/81:40
Your First Computer	Sybex Inc.	2/81:20

Hardware

Product	Manufacturer	Issue/Page
Centronics 739 Printer	Centronics Corp.	12/81:75
Coosol 101B-80E Printer	Coosol Inc.	12/81:76
Dip-81	Dip Inc.	10/81:38
Dynatyper Keyboard Actuator	Rochester Data Inc.	12/81:70
Home Computer Work Table	Computer Roomers Inc.	10/81:39
Internal Memory, The	Holmes Engineering	12/81:68
KGS Keyoard Actuator	Nik International Trading Co.	9/81:26
LX-80	Lobo Drives International	7/81:86
Lynx	Emtrol Systems Inc.	9/81:134
Microconnection, The	The Micro Peripheral Corp.	8/81:154

Product	*Manufacturer*	*Issue/Page*
Microconnection,The	The Micro Peripheral Corp.	9/81:134
Model I Speed-Up	Archbold Electronics	7/81:236
Rex-80	Palomar Computer Products	12/81:68
SA800	Lobo Drives International	7/81:86
Telephone Interface II	Tandy/Radio Shack	9/81:134

Records

Product	*Manufacturer*	*Issue/Page*
Computerworld/Kraftwerk	Warner Bros.	12/81:78

Software

Product	*Manufacturer*	*Issue/Page*
ACCEL2 Compiler	Allan Gelder Software	11/81:38
Acquire	Avalon Hill	11/81:37
Aids II/Aids III	Meta Technologies Corp.	4/81:34
Aids III with Calcs III	Meta Technologies Corp.	8/81:222
Astro-Scope/Sexoscope	Ags Software	9/81:32
Astrology	Tandy/Radio Shack	5/81:32
B1 Bomber/Nukewar	Avalon Hill	6/81:34
Bascom: Basic Compiler, Mod I	Microsoft	5/81:31
Boss	Soft Sector Marketing Inc.	7/81:27
C-Bug	The Micro Works	9/81:29
CCA Data Manager	Personal Software	8/81:222
Cocobug	Allan Gelder Software	9/81:30
Creator, The	Complete Business Systems Inc.	11/81:36
Dancing Demon	Tandy/Radio Shack	4/81:40
Datestones of Ryn	Automated Simulations	4/81:37
DDT (Disk Drive Timer)	Disco-Tech Microcomputer Prod.	4/81:35
Deathmaze 5000	Med Systems Software	2/81:21
Deluxe Personal Finance	Small Systems Group	5/81:37
Discat/Mod I 32K/48K	Myatte Smith	7/81:28
Edas	Misosys	11/81:44
EDTASM Plus	Microsoft	10/81:32
Electra Sketch	Macrotronics Inc.	1/81:44
Eliza	Tandy/Radio Shack	4/81:38
Everest Explorer	Acorn Software Products Inc.	8/81:30
File Management System (FMS)	Exatron Corp.	3/81:19
Flextext	Apparat	9/81:290
Flopycat/Bas, Disk Name/Bas	Marvin Plunkett	5/81:31
Fractional Sound	The Innovative Penguin	9/81:34
Galactic Empire	Broderbund Software	8/81:39
Galaxy Invasion	Big Five Software Co.	8/81:35
Gambiet 80	Microtrend U.S.A.	12/81:80
General Accounting Package	Microed	6/81:26
Haunted House	Tandy/Radio Shack	8/81:36
Hellfire Warrior	Automated Simulations	10/81:28
In Memory Information	Tandy/Radio Shack	11/81:40
Invaders Plus	Level IV Products	8/81:35
Invaders Plus	Level IV Products	4/81:40
Invasion from Outer Space	The Software Exchange	8/81:35
Keynote	Hayden Book Co.	5/81:26
Kim's Game	General Computer Co.	5/81:32
Labyrinth	Med Systems Software	8/81:26
Lazy Writer	ABC Sales	6/81:20
LDOS	Lobo Drives International	6/81:130
Master Directory 1.1	Micro Systems Software	4/81:32
Maxi Micro Manager	Exador/Adventure Intnl.	8/81:222
Midway Campaign	Avalon Hill	6/81:30
Music Box, The	Newtech Computer Systems	5/81:27
Music Master	Instant Software	5/81:26

Product	Manufacturer	Issue/Page
Mychess	Programma International	12/81:80
Orchestra 80	Software Affair	5/81:30
Rats Revenge	Med Systems Software	11/81:40
Regression II	Dynacomp	9/81:28
Remassem 1	Remsoft Inc.	12/81:76
Remodel & Proload, Infinite Basic	Racet Computes	9/81:168
Sargon II	Hayden Book Co.	12/81:80
SFinks	William A. Fink	12/81:80
Small Home/Business Software	Blechman Enterprises	11/81:24
Smart80D	The Micro Peripheral Corp.	9/81:180
Southeastern Textan	Southeastern Software	10/81:33
Space Shuttle	Instant Software	9/81:36
Spooler	Mumford Micro Systems	10/81:30
ST80-III	Small Business Systems Group	9/81:180
Subedit/Subscript	Prosoft Software	6/81:20
Subscript 5.0	Prosoft	9/81:290
Supernova	Big Five Software Co.	8/81:32
Superscript	Acorn Software Products	3/81:20
Superstep	Allen Gelder Software	9/81:34
Swamp Wars	Instant Software	8/81:36
T80-FS1 Flight Simulator	Sublogic Company	8/81:26
The Patch	Cecdat	2/81:22
Tiny Pascal	Supersoft	7/81:168
Trackcess	The Alternate Source	10/81:26
TRS-Opera	Acorn Software	8/81:38
Two Heads of the Coin	Adventure International	8/81:38
Ultra-Mon	Interpro	4/81:37
Varkeep	Circle J Software Ranch	4/81:33
Winged Samurai	Discovery Games	8/81:40
Wizard, The	Programs Unlimited	5/81:33
Word Challenge	Acorn Software	8/81:38
Wordo	Microfantastic Programming	4/81:35
XBasic	Snapp Inc.	9/81:278
XBE	Computer Applications Unlimited	11/81:42
ZBasic Complier	Simutek	6/81:30

80 Reviews 1982 Through October

Hardware

Product	Manufacturer	Issue/Page
AD-111 Audio Amplifier	Thomas Engineering/TCS	2/82:316
AJ 841 I/O Terminal	Anderson-Jacobson	4/82:56
AN-551 Eprom Programmer	Design Solutions Inc.	6/82:68
Base2 850	Base2	4/82:278
C. Itoh Daisy Wheel Printer	C. Itoh Electronics	2/82:316
C. Itoh 8510 Printer	C. Itoh Electronics	5/82:184
Centronics 737	Centronics	4/82:278
Cheaptalk	Alan Saville	5/82:54
Color Computer Disk System	Tandy/Radio Shack	9/82:72
Color Computer Disk System	Tandy/Radio Shack	6/82:48
CPrint Parallel Printer Interface	Micro Labs Inc.	3/82:56
Disk 80	Micromint Inc.	3/82:62
EPS-80 Word Processor	PMC Inc.	9/82:78
Epson MX-80	Epson of America	4/82:278

Product	*Manufacturer*	*Issue/Page*
Escon Selectric Interface	Escon Products Inc.	6/82:47
ETF-80 Electric Typing Fingers	PMC Inc.	9/82:78
Floppy Disk Controller	A.M. Electronics	4/82:53
Gold Plug 80	E.A.P. Co.	5/82:50
Hayes Stack Smartmodem	Hayes Microcomputer Products	10/82:47
IDS 445 Paper Tiger	Integral Data Systems	4/82:278
IDS 460 Printer	Integral Data Systems	4/82:54
Line Printer VII	Tandy/Radio Shack	4/82:82
Line Printer VIII	Tandy/Radio Shack	4/82:70
LNW-80	LNW Research	5/82:206
MDX-2	Micro-Design	4/82:58
MDX-3 Interface PC Board	Micro-Design	10/82:60
Memory Box, The	Displayed Video	5/82:224
Micro Cue Print Buffer	Microcompatible	3/82:58
Micro Mainframe Disk Controller	Micro Mainframe	11/82:58
Microline 80	Okidata	4/82:278
Microline 82A Printer	Okidata Corp.	5/82:56
Micromouth	Micromint	4/82:59
Modem 80	Icrom Enterprises Ltd	6/82:45
Modem I	Tandy/Radio Shack	1/82:51
Novell Image 800 Printer	Novell	5/82:57
Patch 2.0	Cecdat Inc.	6/82:62
P180C Parallel Printer Interface	The Micro Works	6/82:46
Plotter Printer	Tandy/Radio Shack	1/82:122
Plug 'N Power Controller	Tandy/Radio Shack	3/82:61
Power Pack ROM Package	Computerware	6/82:118
Printer/Cassette Interface/PC	Tandy/Radio Shack	4/82:60
QuCeS Hard Disk Subsystem	Quality Computer Services	10/82:154
Selectronic 841	Anderson-Jacobson	4/82:56
SK-2	Mumford Micro Systems	2/82:180
Snapp Spooler	Snapp Inc.	9/82:80
Sooperspooler	Compulink	9/82:66
Sprinter II	Holmes Engineering	11/82:56
Starbuck 8882	Starbuck Data Co.	8/82:42
Strobe 100 Drum Plotter	Strobe Inc.	9/82:58
Tandy 6-Pen Plotter	Tandy/Radio Shack	10/82:58
TP-1 Smith-Corona Printer	Smith-Corona	11/82:61
TRZ-80		1/82:218
UPI-3, UPI-4 Serial Printer Interface	Speedway Electronics	2/82:314
Video 1	Archbold Electronics	5/82:54

Books

Product	*Manufacturer*	*Issue/Page*
55 Color Computer Programs	Arcsoft Publishers	9/82:64
6809 Assembly-Language Programming	Osborne/McGraw Hill	3/82:62
8080/Z80 Assembly-Language Techniques	John Wiley & Sons	2/82:310
Basic 2nd Edition	John Wiley & Sons	2/82:310
Basic Adventures	80-NW Publishing	8/82:37
Basic Conversions Handbook	Hayden Books	9/82:66
Basic Faster and Better	IJG Inc.	5/82:37
Basic for Home Computers	John Wiley & Sons	2/82:310
Basic Handbook, The	Compusoft Publishing	2/82:312
Computer Programming Basic Self Instruction	Cambridge Learning Enterprises	4/82:63
CP/M Primer	Howard W. Sams & Co. Inc.	4/82:65
Data File Programming/Basic	John Wiley & Sons	2/82:310
Graphic Software/Micros	Kern Publications	5/82:37
Intro/8080/8085 Assembly Language Programming	John Wiley & Sons	2/82:310
Learning with Computers	Digital Press	2/82:310
Microcomputer Math	Micro Applications Publishers	3/82:64
Microsoft Basic Decoded	IJG	1/82:38
Murder in the Mansion	Arcsoft Publishers	2/82:326
Pascal	TAB Books	1/82:27

Product	*Manufacturer*	*Issue/Page*
Pascal Handbook	Sybex	1/82:42
Pascal Primer	Howard W. Sams & Co. Inc.	1/82:40
Robot Intelligence/Experiments	TAB Books	6/82:51
RS Model III Service Manual	Tandy/Radio Shack	3/82:65
TRS-80 Graphics	Tandy/Radio Shack	3/82:63
Using CP/M	John Wiley & Sons	2/82:310
Videotext, Coming Revolution	Knowledge Industry Publications	1/82:36

Software

Product	*Manufacturer*	*Issue/Page*
ACCEL2	Allen Gelder Software	5/82:218
Ace Automated Communications	Ace Computer Products	10/82:47
Aircraft Instrument Approach	J.C. Sprott	2/82:318
Alcatraz II	Spectral Associates	8/82:40
Alien Defense	Soft Sector Marketing	8/82:50
Ann Rose	Sturdivant and Dunn	5/82:42
Ants	Synergistic Solar Inc.	4/82:46
Arcade 80	Datasoft Inc.	2/82:320
Armored Patrol	Adventure International	8/82:36
ASPTCH 3.2	Byte Miser Software	2/82:322
Asylum	Med Systems Software	5/82:41
Attack Force	Alpha Products	1/82:51
Auto-Writer	Software Options	9/82:110
Autobasic	Schneider Enterprises	9/82:82
Basic Compiler (BASCOM)	Microsoft Consumer Products	5/82:218
Basicpro	Softworx Inc.	1/82:49
C. C. Writer	Transformation Technologies	3/82:50
Cassette Comm	Tandy/Radio Shack	1/82:44
Cave Hunter	Mark Data Products	9/82:76
Check Register Plus	Softco	6/82:338
Checkwriter 80	Tandy/Radio Shack	6/82:338
Chextext	Apparat	9/82:119
Chextext	Apparat	4/82:98
Chromasette Magazine	CLOAD Magazine	3/82:52
Codes II	Nelson Software	9/82:96
Color Assembler	Computerware	6/82:116
Color Berserk	Mark Data Products	9/82:76
Color Editor	Computerware	6/82:116
Color Pac Attack	Computerware	8/82:45
Color Scripsit	Tandy/Radio Shack	10/82:40
Color Space Invaders	Spectral Associates	3/82:48
Commbat	Adventure International	1/82:43
Compiler Basic, I & III	Tandy/Radio Shack	2/82:164
Compiler Basic, II	Tandy/Radio Shack	2/82:148
Computer Downs	Ernest H. Fellows	2/82:327
Computer Major League	Avalon Hill	5/82:38
Conflict 2500	Avalon Hill	8/82:46
Copyart	Simutek Computer Products	9/82:108
Crush, Crumble, and Chomp	Epyx, Automated Simulations	2/82:324
Data Ace	Computer Software Design	10/82:266
Datahandler	Miller Microcomputer Services	6/82:150
DBase II	Ashton-Tate	6/82:53
Defense Command	Big Five Software	8/82:50
Direct File Transfer/DFT	Big Systems Software	6/82:66
DOSPLUS 3.4	Micro Systems Software	10/82:244
Dragons of Hong Kong	Adventure International	8/82:52
Draw5	Wilson Software Division	1/82:42
DSMBLR	Misosys	9/82:85
Dynamic Report Generator	Dynamic Software	10/82:56
Edit	Southern Software	6/82:71
Electric Pencil 2.0	IJG Inc.	9/82:98
Electric Spreadsheet	Exatron	1/82:49
		9/82:116

Product	*Manufacturer*	*Issue/Page*
Electric Webster		
Eliminator	Adventure International	8/82:54
ENHBAS	Cornsoft Group	1/82:320
Faster 1.2	Prosoft	4/82:40
Flextext/80	Apparat	9/82:100
GEAP	J.F. Consulting	10/82:184
Grammatik	Aspen Software	9/82:120
Graphics Editor/Programmer	J.F. Consulting	3/82:57
GRBasic 3.0	Med Systems Software	6/82:54
Hexspell	Hexagon Systems	4/82:98
Hexspell 2	Hexagon Systems	9/82:116
HORKS	Edu-Ware East	9/82:70
ICL	XYZT Computer Dimensions	6/82:64
IDM/V	Micro Architect Inc.	6/82:150
Institute, The	Med Systems Software	9/82:64
INTASM 2.0	Singular Systems	9/82:68
KWICOS	Kwik Software	2/82:322
Lablmaker	ETS Center	10/82:34
Laser Defense	Med Systems Software	8/82:39
Lazy Writer	Soft Sector Marketing	9/82:102
LDOS 5.1	Logical Systems Inc.	9/82:252
Letter-Writer	Astro-Star Enterprises	9/82:112
Life List	Manhattan Software	5/82:46
Litigation Support System	Tandy/Radio Shack	6/82:50
Local Call for Death	Adventure International	8/82:52
LZ Patcher	Imaginuity Inc.	9/82:104
Maces & Magic	Adventure Int./Chameleon	8/82:38
Macro-Mon, The Shadow	Advanced Operating Systems	9/82:292
Mailmerge	Micropro	9/82:94
Master Control	Soft Sector Marketing	3/82:49
Maxi Check Register Acct.	Exador/Adventure Intnl.	6/82:338
Maxi Manager 3.1	Adventure International	6/82:150
Maxprint	Peggytronics	10/82:33
Meteoroids	Spectral Associates	3/82:48
Micro-Typit	Coolidge & Associates	9/82:114
Microcosm III	Basics & Beyond Inc.	1/82:48
Microproof	Cornucopia Software	9/82:116
Microproof	Cornucopia Software	4/82:98
Microstat	Ecosoft	6/82:280
Mind Thrust	Hayden Software	8/82:56
Miz' Spell	Programs Unlimited	4/82:98
Miz' Spell	Programs Unlimited	9/82:120
Mod II Game Package	Small Business Systems	5/82:52
MZAL	Computer Applications Inc.	10/82:160
New Starship Battles	Synergistic Solar Inc.	4/82:46
NEWDOS 80 V. II	Apparat	2/82:152
NewScript	Prosoft	9/82:104
NewScript 6.0	Prosoft	3/82:53
NewScript 7.0	Prosoft	10/82:52
Newtalk	Star Kits	10/82:38
Newtrieve	Unique Printing & Stationery	10/82:42
Omniterm	Lindberg Systems	4/82:94
On Her Majesty's Ship Impetuous	Adventure International	8/82:52
Packer 1.5	Cottage Software	6/82:60
Parsector 8	Synergistic Solar Inc.	4/82:46
Pencil Plus	Dan Robinson	9/82:96
Penetrator	Melbourne House Software	9/82:74
Pensawrite 1	Pensadyne Computer Services	9/82:112
Pensawrite 2	Pensadyne Computer Services	9/82:106
Personal Finance for C.C.	Tandy/Radio Shack	10/82:38
PL/I-80	Digital Research	6/82:122
Powerdraw	Powersoft	10/82:40
Profile II	Tandy/Radio Shack	4/82:48

Product	*Manufacturer*	*Issue/Page*
Profile III Plus	Small Computer/Tandy	10/82:308
Profile Plus	Tandy/Radio Shack	4/82:48
Proofreader	Aspen Software	4/82:98
Proofreader/Proof-Edit	Aspen Software	9/82:118
QSD Utility Disk #1	Quality Software Distributors	4/82:41
Qwerty	Med Systems Software	10/82:34
Qwerty	Med Systems Software	9/82:102
Refware Thesaurus	David C. Whitney Associates	4/82:52
Robot Attack	Big 5 Software	8/82:39
RPM	Prosoft	4/82:40
Scarfman	The Cornsoft Group	3/82:53
Scrinput	ACR Consultants	4/82:44
Scriplus	Quality Software Distributors	9/82:101
Scripmod	MG Products	9/82:102
Scripsit	Tandy/Radio Shack	9/82:98
Scripsit Dictionary	Tandy/Radio Shack	9/82:120
Scripsit, Mod II	Tandy/Radio Shack	2/82:222
Scriptr	Pioneer Software	9/82:102
SDS80C Editor/Assem/Mon	The Micro Works	6/82:70
Search and Sort	Micro Pro Systems	8/82:56
SECS	Datasoft Inc.	3/82:52
SFinks 3.0	William Fink	9/82:80
Sigmon	Datasoft Inc.	3/82:52
Smart Terminal Program, STERM	Howe Software	6/82:60
Snapp III Extended Built-In Functions	Snappware	1/82:134
Snapp IV Auto Map	Snappware	1/82:134
Snapp V Extended File Map	Snappware	5/82:216
Softext/Softscreen	Aspen Software	9/82:106
Space Ace 21	Synergistic Solar Inc.	4/82:46
Space War	Spectral Associates	3/82:48
Space Waste Race	Storybooks of the Future	5/82:42
Special Delivery	Software Concepts	9/82:110
SpellGuard	Innovative Software Applications	9/82:95
SpellStar	Micropro International	9/82:95
Star Blaster	Micro Works	8/82:44
Star Warrior	Automated Simulations	4/82:50
Starfighter	Adventure International	2/82:320
Statistical Analysis	Tandy/Radio Shack	6/82:280
Stud5	Wilson Software Division	1/82:42
Super Color Writer 1.0	Nelson Software	9/82:62
Super Micro Pro Football	Micro Pro Systems	6/82:54
Superscript	Acorn Software	9/82:101
Supersort	Micropro International	9/82:94
Tanktics	Avalon Hill	4/82:63
Tape Downloading System/TDS	Big Systems Software	6/82:66
TASMON	Alternate Source	1/82:126
Telewriter	Cognitec	5/82:208
Temple of Apshai	Automated Simulations	2/82:323
Tower of Orlandor	Compu-Things	8/82:46
Two Heads of the Coin	Adventure International	8/82:52
Uniterm	B.T. Enterprises	6/82:56
Utility Pack, The	B.T. Enterprises	4/82:42
VisiGraph	Micro Software Systems	10/82:54
Voyage of the Valkyrie	Advanced Operating Systems	5/82:44
Word IV/Word V	Micro Architect	9/82:110
Word Machine, The	GB Associates	9/82:112
Wordslinger, The	Instant Software	9/82:114
Wordsmith	ABS Suppliers	6/82:63
Wordsmith	ABS Suppliers	9/82:112
WordStar	Micropro International	9/82:92
XBug	Computer Applications Unlimited	3/82:54
XTEND40	Prosoft	4/82:40
ZBasic 2.0	Simutek Computer Products	5/82:48

User Groups

This magazine is not the only good information source for your TRS-80. There are many computer clubs, called user groups, that help out members with technical tips and a pool of experience.

Below is a list of user groups that cater to the TRS-80 owner, many exclusively. The list is long, but by the time you read it, there will certainly be more. Look for one in your area and drop them a line.

If you belong to a user group that does not appear on this list, send a letter to *80 Micro* telling us about the club. We'll get your group in our next list. We also want to hear about any updates on the groups in this list. ■

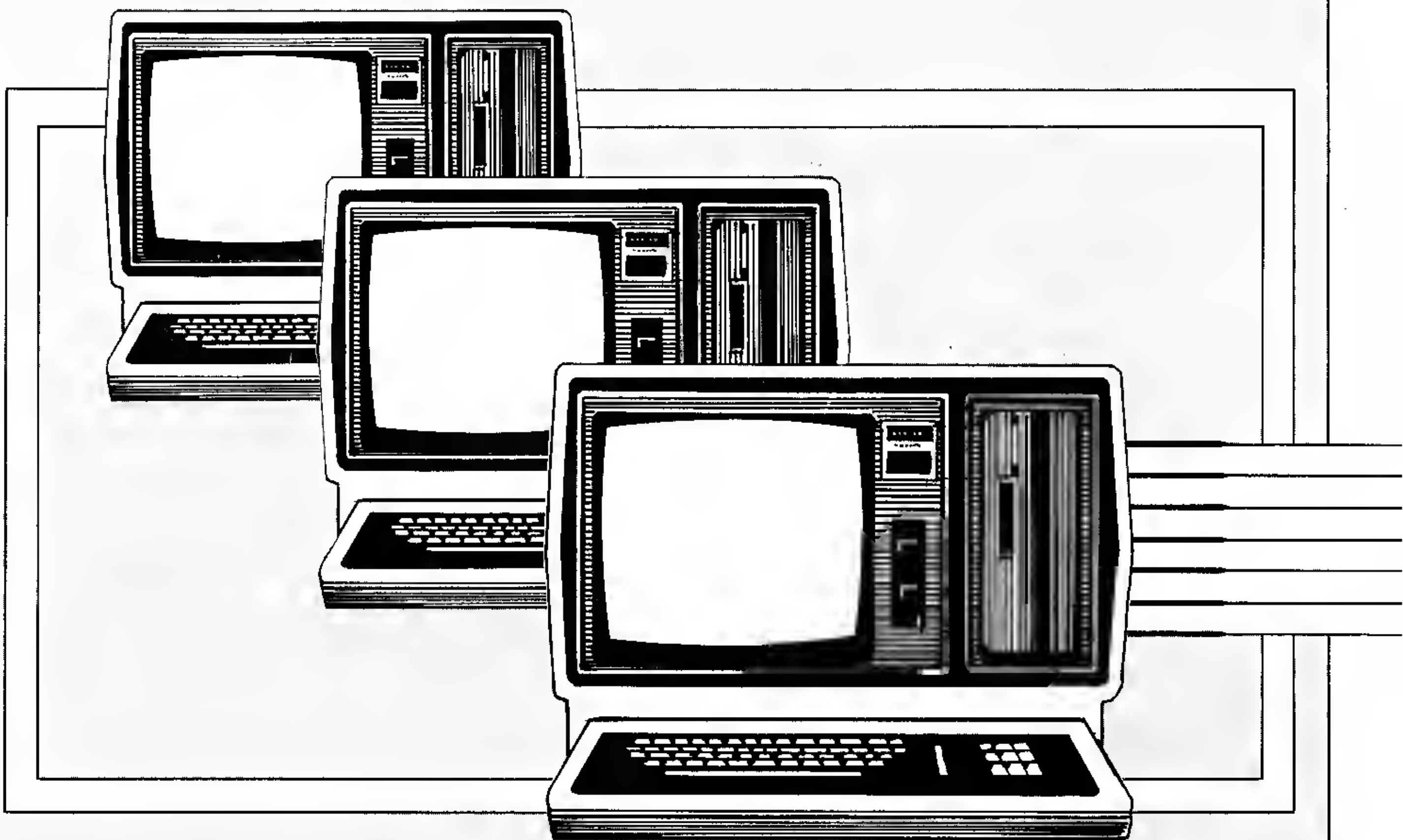
STATE	GROUP NAME	ADDRESS	CITY	ZIP
AK	ANCHORAGE USERS GROUP	BOX 10, 385 SOUTH STATION	ANCHORAGE	99511
AL	CENTRAL AL MICRO SOCIETY JUNIOR CHAPTER	RT. #3 BOX 570	MONTGOMERY	36110
AL	CENTRAL ALABAMA COMPUTER SOCIETY	2073 REXFORD RD	MONTGOMERY	36116
AL	G2C3	4307 OLD SHELL RD	MOBILE	36608
AZ	PHEONIX USERS GROUP	1850 EAST MARYLAND #27	PHEONIX	
AZ	ARIZONA USERS GROUP	4322 EAST FAIRMONT	PHOENIX	85018
AZ	USERS GROUP OF ARIZONA	6218 W. MARLETTE	GLENDALE	85301
BRA	BRASILIAN CLUB	RUA SAMBAIBA 516, LEBLON	RIO DE JANEIRO	22450
CA	THE FORTH INTEREST GROUP	PO BOX 1105	SAN CARLOS	94070
CA	HOMEBREW COMPUTER CLUB	BOX 626	MOUNTAIN VIEW	94042
CA	VENTURA COUNTY TRS-80	2534 NORTH TEMPLE AVE.	CAMARILLO	93010
CA	USERS GROUP	7465 HOLLISTON AVE., SUITE 23	GOLETA	93017
CA	USERS GROUP	712-C COUNTRY WOOD	WALNUT CREEK	94598
CA	USERS AND ABUSERS	1350 GRANT RD.	SUNNYVALE	94040
CA	REDWOOD EMPIRE USERS GROUP	7136 BELITA AVE.	ROHNERT PARK	94928.
CA	SOUTH BAY USERS GROUP	3605 PINE AVE	MANHATTEN BEACH	90266
CA	EAST BAY USERS GROUP	17 ECHO AVE	OAKLAND	94611
CA	SAN FRANCISCO TRS-80 USERS GROUP	338 ALIDA WAY #306	SOUTH SAN FRANCISCO	94080
CA	CAUSE (SOFTWARE EXCHANGE)	18651 VON KARMAN	IRVINE	92713
CA	INLAND COMPUTER SOCIETY	3359 SECOND ST	RIVERSIDE	92501
CA	SOUTH BAY USERS GROUP	BOX 6302	STANFORD	94305
CA	TRS-80 FREE PROGRAM EXCHANGE	4418 MORROW RD.	MODESTO	95350
CA	COUNTRYWIDE USER GROUP	10409 STATE ST.	SOUTH GATE	90280
CA	LITTON CALULATOR/COMPUTER CLUB	5500 CANOGA AVE	WOODLAND HILLS	91364
CA	MARIN COUNTY USERS GROUP (MCTUG)	45 SELFRIDGE WAY	HAMILTON AFB	94934
CA	SONOMA COUNTY COMPUTER CLUB	BOX 945	COTATI	94928
CA	TRS-80 NIBBLERS	2555 HESPERIAN BLVD	HAYWARD	94545
CA	MONTEREY.BAY USERS GROUP	1002 HALSEY DR	MONTEREY	93940
CA	VALLEY COMPUTER CLUB	3311 WEST THIRD, APT. 1-319	LOS ANGELES	90020
CA	PAMONA COMPUTER SOCIETY	4155 OAK HOLLOW RD.	CLAREMONT	91711
CA	COMPUTER INFO EXCHANGE (CIE SOFTW NEWS)	BOX 159	SAN LUIS REY	92068
CA	ORANGE COUNTY USERS GROUP	3521 E. COMMONWEALTH	FULLERTON	92631
CA	ET-3400 USERS GROUP	11231 OAK ST.	EL MONTE	91731
CA	TRS-80 USERS GROUP OF NAPA	4432 SPRINGWOOD	NAPA	94558
CA	OASIS USERS GROUP	PHASE ONE SYSTEMS	OAKLAND	
CA	VENTURA COUNTY TRS-80 COMPUTER CLUB	567 WEST LOOP DR.	CAMARILLO	93030
CA	VALLEY TRS-80 USERS GROUP (VTUG)	19116 NASVILLE ST	NORTHRIDGE	91326
CA	SACREMENTO USERS GROUP	1237 BEARD WAY	CARMICHAEL	95608
CA	SAN JOSE USERS GROUP	3490 BON AIR COURT	SAN JOSE	95117
CA	SAN FRANCISCO USERS GROUP	HQ USARRIX	PRESIDIO OF SAN FRAN	94129
CA	FORTH INTEREST GROUP	PO BOX 1105	SAN CARLOS	94070
CA	SAN GABRIEL VALLEY TRS-80 USERS GROUP	750 E. 5TH ST. #75	AZUSA	91702
CA	SILICON VALLEY COMPUTER CLUB	P.O. BOX 61593	SUNNYVALE	94088
CAN	KITCHENER-WATERLOO MICRO CLUB	UNIVERSITY OF WATERLOO	WATERLOO, ONT	N2L3G1
CAN	MICRO-80 COMPUTER CLUB OF OTTAWA	178 MONTEREY DR	NEPEAN, ONTARIO	K2H7A8
CAN	REGINA OPERATORS OF MICRO SYSTEMS	BOX 1001	REGINA	S4P3B2
CAN	TMUG, TORONTO MICRCOMPUTER USERS GROUP	OX 857, STATION A	TORONTO, ONT	M5W1G3
CAN	COLOR COMPUTER USERS GROUP	33 KILLDEER CRESCENT	TORONTO, ONTARIO	
CAN	AURORA COMPUTER SOCIETY	BOX 4342	EDMONTON, ALBERTA	T5H1R5
CAN	INTERESS-80	673 CLAIRETTE	FABREVILLE, QUEBEC	H7P2Y3
CT	NORTH EASTERN BASIC FOUR USER GROUP	22 TOBEY RD.	BLOOMFIELD	06002
CT	FAIRFIELD COUNTY TRS-80 USERS GROUP	10 RICHLEE RD.	NORWALK	06851
CT	SOUTHERN NEW ENGLAND COMPUTER SOCIETY	267 WILLOW ST	NEW HAVEN	06511
CT	CONN MICROISTS	8802 WENDY LANE	WESTPORT	06881
CT	CONNECTICUT COMPUTER CLUB	18 RIDGE COURT WEST	WEST HAVEN	06516
DE	DELAWARE USERS GROUP	1116 PIPER RD	WILMINGTON	19803
DE	DELAWARE USERS OF MICROS	318B CHAPEL AVE	CLAYMONT	19703
DE	SOUTHEASTERN DELAWARE CTY TRS80 USR GRP	1109 MADISON AVE	PROSPECT PARK	19076
ENG	NATIONAL TRS-80 USERS GROUP	40A HIGH ST., STONY STRATFORD	MILTON KEYNES	
FL	TBUG-80 NEWSLETTER (TAMPA BAY GROUP)	322 S. 21ST STREET	HAINES CITY	33844
FL	TBUG-80	PO BOX 247	TAMPA	33602

STATE	GROUP NAME	ADDRESS	CITY	ZIP
FL	ORLANDO, FL, COMPUTER CLUB	ADVENTURE INTERNATIONAL	ORLANDO	32802
FL	CLUB OF CENTRAL FLORIDA	152 MILL RUN DRIVE	LAKE MARY	32746
FL	FL EDUCATIONAL COMPUTER USERS GROUP	5720-1 ATLANTIC BLVD.	JACKSONVILLE	32207
FL	TAMPA BAY USERS GROUP	322 S 21ST ST	HAINES CITY	33844
GA	USERS GROUP OF ATLANTA, LTD.	3408 CLAIRMONT RD, N.E.	ATLANTA	30319
GA	STATESBORO TRS-80 USERS GROUP	223 N. EDGEWOOD DRIVE	STATESBORO	30458
GA	CSRA COMPUTER CLUB	BOX 284	AUGUSTA	30903
GA	USERS GROUP OF ATLANTA, LTD.	1315 RUSTIC RIDGE DR, N.E.	ATLANTA	30319
GRC	USERS GROUP	CHRISSOSTOMOU SMYRNIS 30	NEO PSYHIKO	
HOL	TRS-80 GEBRUIKERS VERENIGING BENELUX	EIZENLAAN 8, 1214 KL	HILVERSUM	
IA	TRS-80 USER GROUP	1400 EAST POST B	MARION	
IA	MARSHALLTOWN COMPUTER CLUB	1101 SOUTH 2ND AVE	MARSHALLTOWN	50158
IA	DES MOINES USERS GROUP (DM-TUG)	#303 4215 GRAND AVE	DES MOINES	50312
IL	COMPUTER RENTAL SERVICE	RR BOX 1138	CRETE	60417
IL	PEORIA AREA COMPUTER CLUB	2019 NORTH IDAHO	PEORIA	61604
IL	CHICAGO TRS-80 USERS GROUP	3950 N.LAKE SHORE DRIVE	CHICAGO	60613
IL	QUAD CITY COMPUTER CLUB	4211 1/2 7TH AVE.	ROCK ISLAND	61201
IL	CHICAGO AREA COMPUTER HOBBYIST EXCH	BOX 52	SOUTH HOLLAND	60473
IL	ONE ON ONE	948 PROSPECT AVENUE	ELMHURST	60126
IN	INDIANAPOLIS TRS-80 USERS GROUP	2203 CORD ST.	SPEEDWAY	46224
IN	GROUP OFF SOUTHWEST INDIANA	BOX 3284	EVANSVILLE	47732
KS	COMPUTER NETWORK OF KANSAS CITY	7631 BROADMOOR LANE	OVERLAND	66204
KS	KBUG	BOX 1398	WICHITA	67201
KS	GREATER KANSAS CITY USERS GROUP	8909 WENONGA	LEAWOOD	62206
KY	TRS-80 USERS CLUB		LOUISVILLE	
LA	CRESCENT CITY COMPUTER CLUB	PO BOX 1097, U. OF NEW ORLEAN	NEW ORLEANS	70122
MA	SMALL BUSINESS SYSTEMS GROUP (SBSG)	6 CARLISLE ROAD	WESTFORD	01886
MA	TRUGEM	61 LAKE SHORE RD	NATICK	01760
MA	NEW ENGLAND SYSTEM/3, INC.	PO BOX 1275	LOWELL	01853
MA	NATIONAL COMPUTER SHOWS	BOX 678	BROOKLINE VILLAGE	02147
MA	BOSTON COMPUTER SOCIETY	17 CHESTNUT ST	BOSTON	02108
MA	PRIME COMPUTER, INC.	40 WALNUT ST.	WELLESLEY	02181
MA	GOSUB TRS80 USERS GROUP	PO BOX 712	WORCESTER	01613
MA	WESTERN MASS COMPUTER CLUB	134 BRECKWOOD CIR.	SPRINGFIELD	01119
MA	ALCOVE COMPUTER CLUB	230 MAIN ST	NORTH READING	01824
MA	NEW ENGLAND COMPUTER SOCIETY	PO BOX 198	BEDFORD	01730
MA	CULPRIT/EDP-AUDITOR USER GROUP	20 WILLIAM ST.	WELLESLEY	02181
MA	TRS-80 NEWSLETTER	96 DOTHAN ST.	ARLINGTON	02174
MD	TRS-80 BALTIMORE USERS GROUP	3505 N. CHARLES ST.	BALTIMORE	21218
ME	AUGUSTA USERS GROUP	BOX 2143	AUGUSTA	04330
ME	SOUTHERN MAINE TRS-80 USERS GROUP	15 MOUNTAIN VIEW ROAD	CAP ELIZABETH	04107
MI	MICROCOMPUTER USERS INTERNATIONAL	1804 WEST 18TH ST	SAULTE STE. MARIE	49783
MI	CENTRAL MICHIGAN USERS GROUP	5582 CORAL WAY	HASLETT	48840
MI	CENTRAL MICHIGAN USERS GROUP	938 WILDEWOOD	EAST LANSING	48823
MI	MACUL (MICH. ASSN. FOR COMPUTER USERS)	C/O WCISD 33500 VAN BORN ROA	WAYNE	48184
MI	COLOR COMPUTER OWNER'S GROUP (CCOG)	P.O. BOX 1113	DEARBORN	48121
MI	EDUCATIONAL RECREATIONAL CLUB (ERCC)	PO BOX 325	OWOSSO	48867
MI	MID-MICHIGAN COMPUTER CLUB	15151 RIPPLE DR.	LINDEN	48451
MI	SOUTHEASTERN MICHIGAN COMPUTER	BOX 9578	DETROIT	48202
MI	MACUL (MICH ASSOC COMP USERS/LEARNING)	33500 VAN BORN RD	WAYNE	48184
MI	HAMUG	HOPE COLLEGE	HOLLAND	49423
MI	FARGO-MOORHEAD COMPUTER CLUB	111 SO MAIN ST.	DILWORTH	56529
MN	MINNESTOTA COMPUTER SOCIETY	BOX 35317	MINNEAPOLIS	55435
MO	ST. LOUIS AREA COMPUTER CLUB	PO BOX 28924	ST. LOUIS	63132
MS	OXFORD Z-80 USERS GROUP	PO BOX 847	UNIVERSITY	38677
NC	NATIONAL COMPUTER SOCIETY	PO BOX 41205	FAYETTEVILLE	28304
NC	USERS NOTES	7554 SOUTHGATE RD.	FAYETTEVILLE	28304
NH	NORTHERN NEW ENGLAND SOCIETY	PO BOX 69	BERLIN	03570
NH	GOSUB USERS GROUP	346 S. TAYLOR ST.	MANCHESTER	03103
NJ	AMATEUR COMPUTER CLUB OF NJ	UCTI 1776 RARITAN RD	SCOTCH PLAINS	07076
NJ	TRS-80 USERS GROUP OF CHERRY HILL, NJ	2742 VIRGINIA TRAILS	BROWNS MILLS	08015
NJ	NORTHERN NJ AMATUER COMPUTER CLUB	6 BRYSON ROAD	FAIR LAWN	07410
NJ	BASIC FOUR USER GROUP, MID-ATLANTIC	PO BOX 2214	CLIFTON	07015
NJ	CENTRAL JERSEY COMPUTER CLUB	RD #1, BOX 147	HOPEWELL	08525
NM	LLANO ESTACADA COMPUTER CLUB	1509 FAIRWAY TERRACE	CLOVIS	88101
NM	APPRAISER'S TRS-80 CLUB	1215 FRUIT, NW	ALBUQUERQUE	87102
NV	NORTHERN NEVADA AMATUER COMPUTER CLUB	PO BOX 9068	RENO	89507
NY	TRS-80 COLOR COMPUTER CLUB	347 WEST 48TH ST.	NEW YORK	10036
NY	ZWEIBRUECKEN MICROCOMPUTER USERS GROUP	TAMMC ARMT DIV.	APO NEW YORK	09052
NY	TRS-80 USERS GROUP	245 MAPLEVIEW RD	CHEEKTOWAGA	14225
NY	METRO TRS-80 USERS GROUP	310 WEST 106 ST - 15D	NEW YORK	10025
NY	KINGS BYTE TRS-80 USERS GROUP	1063 EAST 84 ST.	BROOKLYN	11236
NY	USERS GROUP (DIVISION OF C.H.I.P.S.)	407 N. 6TH ST.	FULTON	
NY	TRS-80 SPECIAL INTEREST GROUP	RD 2	BLOSSVALE	13308
NY	TRS-80 USERS GROUP, CHURCH APPLICATIONS	PO BOX 41	MASONVILLE	13804
NY	CENTRAL NEW YORK TRS-80 USERS GROUP	26 JAMESVILLE VE. J-4	SYRACUSE	13210
NY	SAINT PETER'S COMPUTER CLUB	JAMES ST.	ROSENDALE	12472
NY	USERS GROUP	RT #1, BOX 8	MILTON	12547
NY	INTERNATIONAL SOFTWARE A.G. USERS GROUP	1 NEW YORK PLAZA	NEW YORK	10004
NY	CP/M USERS GROUP	164 WEST 83RD ST	NEW YORK	10024
NY	S & N COLOR, I AND III CLUB	3 BOHR COURT	SPRING VALLEY	10977
NY	ONONDAGA COUNTY	BAKER HALL,ELECTRONIC'S PKWY	SYRACUSE	
NY	LONG ISLAND COMPUTER ASSN	36 IRENE LANE E	PLAINVIEW	11803
OH	DAYTON AREA USERS GROUP	231 GREEN ST	DAYTON	45402
OH	CLEVELAND DIGITAL GROUP	P.O. BOX 17440	CLEVELAND	44117
OH	TRS-80 CLUB	8106 QUAILWOOD CT	WEST CHESTER	45069
OH	BASIC FOUR USER GROUP, N. OHIO	3203 W. 71ST ST.	CLEVELAND	44102
OH	NORTH CENTRAL OHIO COMPUTER SOCIETY	P.O. BOX 965	MANSFIELD	44901
OH	PRIDE USERS ASSOCIATION, INC.	1248 SPRINGFIELD PIKE	CINCINNATI	45215

STATE	GROUP NAME	ADDRESS	CITY	ZIP
OH	TRS-80 ACSCO (ADVENTURE COMP. SOCIETY)	BOX 28355	COLUMBUS	43228
OK	TULSA USERS GROUP	PO BOX 1133	TULSA	74101
OR	PORTLAND USERS GROUP	13485 S.W. DRIFTWOOD	BEAVERTON	97213
OR	EUGENE TRS-80 USERS GROUP	296 HERITAGE AVE	EUGENE	97404
OR	TEKTRONIX, INC.	PO BOX 500	BEAVERTON	97077
OR	CENTER FOR THE STUDY OF THE FUTURE	4110 NE ALMEDA	PORTLAND	97212
PA	CRANBERRY COMPUTER CLUB	RADIO SHACK STORE, CRANBERRY	CRANBERRY	16319
PA	LEHIGH VALLEY COMPUTER GROUP	P.O. BOX 2952	LEHIGH VALLEY	18001
PA	DELAWARE VALLEY COMPUTER CLUB	BOX 651	LEVITTOWN	19058
PA	PHILADELPHIA AREA COMPUTER SOCIETY	PO BOX 432	AUDUBON	19403
PA	AUUA, INC.	FLEET MATERIAL SUPPORT OFF.,	MECHANICSBURG	17055
PA	PITTSBURGH AREA COMPUTER CLUB	400 SMITHFIELD ST.	PITTSBURGH	15222
PA	CAPATUG (CAPITAL AREA TRS80 USERS GRP)	4644 CARLISLE PIKE	MECHANICSBURG	17055
PA	ATLANTIC SOFTWARE, INC.	901 LAFAYETTE BLDG., 5TH & CH	PHILADELPHIA	19106
PR	SPANTRASH CLUB 80	P.O. BOX 9475	PONCE	00732
RI	RI TANDY USERS GROUP (RITUG)	1441 PARK AVE.	CRANSTON	02920
RSF	TRS-80 CLUB	PO BOX 35461, NORTHCLIFF 2115	REP S AFRICA	23.881
TN	CHATTANOOGA MICROCOMPUTER CLUB	4429 PAULA LANE	RED BANK	37415
TN	NASHVILLE TRS-80 USERS GROUP	BOX 2891	NASHVILLE	37219
TX	FORTRUG	7952 HIGHWAY 80	WEST FORT WORTH	76116
TX	TRS-80 USERS GROUP	PARK ROW AT COLLINS	ARLINGTON	76010
TX	COMPUTER CLUB OF SAN MARCOS	PO BOX 199	SAN MARCOS	78666
TX	NASA/BAY AREA USERS GROUP	PO BOX 57116	WEBSTER	77598
TX	TEXHOMA MICROCOMPUTER ENTHUSIASTS	PO BOX 4391	WICHITA FALLS	76308
TX	PERMIAN BASIN COMPUTER GROUP	BOX 3912	ODESSA	79760
TX	HIGH PLAINS USERS GROUP	PO BOX 30545	AMARILLO	79120
TX	KILLEEN USERS GROUP	BOX 510	KILLEEN	76541
TX	PAN HANDLE COMPUTER SOCIETY	3440 BELL SUITE 226	AMARILLO	79109
TX	USERS GROUP	1516 DALLAS ST.	KILLEEN	76541
TX	WICHITA VALLEY USERS GROUP	PO BOX 4391	WICHITA FALLS	76308
TX	COMPUTER HOBBYIST GROUP OF NORTH TEXAS	BOX 1344	GRAND PRAIRIE	75051
TX	TRS-80 USERS GROUP	7822 SHADY HOLLOW LANE	SAN ANTONIO	78255
TX	EAST TEXAS COLOR COMPUTER CLUB	2101 EAST MAIN	HENDERSON	75652
TX	USERS GROUP OF S.W. HOUSTON	3723 PURDUE	HOUSTON	77005
TX	ALAMO COMPUTER ENTHUSIASTS	4847 CASTLE SHIELD	SAN ANTONIO	78218
TX	TEXAS USERS GROUP	5224 WINIFRED DR	FT. WORTH	76133
UK	NORTH LONDON HOBBY COMPUTER CLUB	HOLLOWAY RD.	N. LONDON	N7 8DB
UK	UK LEVEL I USERS GROUP	123 ROUGHWOOD DRIVE	NORTHWOOD KIRKBY MER	L339UG
VA	RICHMOND TRS-80 USERS' GROUP	C/O MC STUDIO, 4115 HOPKINS R	RICHMOND	23234
VA	TIDEWATER TRS-80 USERS GROUP	BOX 10281	NORFOLK	23513
VA	PRINCE WILLIAM COMPUTER CLUB		WOODBRIDGE	
VA	AMATEUR RADIO REASEARCH AND DEV. CORP	1524 SPRINGVALE AVE	MCLEAN	22101
VA	TCUG, INC	PO BOX 2826	FAIRFAX	22031
VT	NORTHERN VERMONT COMPUTER CLUB		ROXBURY	05669
WA	USERS GROUP	SOUTH 3718 MANITO BLVD.	SPOKANE	99203
WA	USERS CLUB--WASHINGTON	PO BOX 7112	TACOMA	98407
WA	USERS GROUP	3825 NORTH 26TH ST.	TACOMA	98400
WA	NATIONAL CINCOM USERS GROUP	480 HOUSER WAY NORTH	RENTON	98055
WA	USERS GROUP OF SPOKANE	S. 3718 MANITO BLVD	SPOKANE	99203
WA	USERS GROUP - WASHINGTON	21814 PACIFIC HWY SOUTH - LOT	DES MOINES	98188
WGR	TUG	1M ORTFELD 12	4300 ESSEN 14	
WI	TRI-CO GSD USERS	PO BOX 768	MANITOWOC	54220
WI	USERS NOTES	61 13TH AVE. SOUTH	ONALASKA	54650
WI	MADISON USERS GROUP	430 JEAN ST	MADISON	53703

Part IV.

MODEL II



The Art of Encoding and Decoding

by Karl Andreassen

Use your computer to translate your messages into an indecipherable mess; then, with this program, you can translate them back again.

The work of encrypting and analyzing encrypted messages is drudgery, even though the results are fascinating and the challenge of a cipher message too intriguing to pass up. This drudgery was recognized centuries ago, and machines were invented to create and decode messages.

The art and craft of cryptanalysis has been neglected because many assumed that ciphers were impossible to break.

It wasn't until World War I that cryptology came into its own, and an agency of the War Department was formed specifically for making and breaking codes.

A$( )	Straight alphabet
B$( )	Inverted alphabet
C$( )	Accumulate ciphertext
A$	Decode key
B$	Encode key
C$	Encode/Decode switch
D$	Hardcopy/Quit switch
E$	Transient variable
Z$	Keyboard string
I	Keyboard count

Table 1

While computers have increased the demand for cryptanalysts, they have relieved the cryptographer of the drudgery of cracking an encrypted message for which he has no key. The computer doesn't do all the work; the cryptanalyst is freed from the trial and error, pencil and paper work formerly required.

Stripped of drudgery by the computer, cryptanalysis becomes a fascinating challenge and a great mind developer. But one thing the computer cannot do: It cannot replace human ingenuity and the all-important hunch. Deductive and inductive reasoning alone are insufficient; the cryptanalyst becomes almost prescient at times.

The Program

This Model II program, Quick Crypto, encodes and decodes messages, using a simple substitution cipher. Line 80 produces a standard A, B, C. . .Z alphabet, and line 90 produces its inversion. Line 150 produces the cipher encoding key, while line 140 reads the decoding key into A$. By adding lines and increasing the program's complexity, the program could have produced the decoding key.

The cipher alphabet is a double alphabet with the reversed alphabet interspersed between every other letter of the straight alphabet. This may be somewhat complicated to those new to cryptology, but it is one of the most elemental cipher alphabets.

To keep the program within bounds, the encoding must be uppercase and without numerals or punctuation. Most cipher messages are sent and received in this form since punctuation would assist the cryptanalyst in cracking the code. Substituted letters are formed into five-letter code groups, this being the average number of random letters that can be easily keyed into the decoding machine.

As you enter the plaintext (the original message for encoding), the computer screens the message exactly as received from the keyboard. If the message exceeds 500 letters, the program jumps to line 420 and screens the coded text directly beneath the plaintext version. If the message is less than 500 letters, pressing uppercase 8 or * accomplishes the same result. Then the option to printout appears. The printout is the enciphered message only, ready for mailing.

If your addressee is privy to this program, it is simple to decode the message. The menu selection EN or DE engages line 250 if you enter DE. Again, you can use the spacebar, but no punctuation or numerals (an error will result if you do). Backing up and striking over also produces an error; if you make a mistake in entering the message, just keep on going or rerun the program.

As in entering plaintext, when the cipher entry is complete, press * and

```
TZGNY  ZQXQI  XFZJV  GWJSY  STHPJ  XIZAP  QXDZK
XAXXG  FSQJA  XYZPQ  XTZGN  QJZJX  QQPHH  SIJJD
XIPFX  JDZJX  FXYJI  STZWG  XJVYV  THPFQ  XQZIX
GSJHI  SHXIJ  NYJDU

MANYCASESCELATINGTOCOMPUTERABUSEHAVEBEEN-
LOSTBECAUSEMANYSTATESSUPPORTTHERULETHAT-
ELECTROMAGNETICIMPULSESARENOTPROPERTYBZND
```

Figure 1

The Key Box

Model II Basic
Model II
64K RAM

the decoded version appears on screen in solid lines without spaces. Punctuation and spaces are almost as great a bounty for a cryptanalyst as having the original cipher key handed to him.

The printout option appears, and the solid lines are repeated on the printer. Punctuation is easy to edit into the completed plaintext, if this is a message shared with others unfamiliar with crypto procedures.

If you and your addressee have access to modems, you can encrypt your outgoing messages and decipher your incoming messages directly, without going through the printout stage.

Upcoming *80 Micro* issues will carry a series on cryptology and cryptanalysis, with programs to assist in decoding with an unknown cipher. ■

Karl Andreassen can be reached at 24750 Chianti Road, Cloverdale, CA 95425.

Program Listing

```
 10 CLS
20 PRINTTAB(30)"BASIC CRYPTO PROGRAM"
30 PRINTTAB(31)"by Karl Andreassen"
40 '==Copyright July 3 1982
50 CLEAR 7500 :I=1
60 DIM A$(26),B$(27),C$(500),D(26)
70 FOR X=1 TO 26
80 A$(X)=CHR$(X+64)
90 B$(X)=CHR$(91-X)
100 NEXT X
110 DATA B,D,F,H,J,L,N,P,R,T,V,X,Z
120 DATA Y,W,U,S,Q,O,M,K,I,G,E,C,A
130 FOR X=1TO 26
140 READ F$ :A$=A$+F$
150 B$=B$+B$(X)+A$(X)
160 NEXT X :PRINT
170 PRINT "TO WRITE A MESSAGE IN
  CIPHER, ENTER < EN >"
180 LINEINPUT "  TO DECODE A CIPHER
  MESSAGE, ENTER < DE >  ";C$
190 IF C$="EN" THEN 210
200 IF C$="DE" THEN 250ELSE 170
210 PRINT :PRINTTAB(15) "Write message
  in plain language, using spacebar
220 PRINTTAB(15) "between words.  No
  punctuation or numbers permitted."
230 PRINTTAB(15) "UPPERCASE letters
  only.  Touch < * > at end of
  message."
240 PRINT :PRINT "BEGIN:" :PRINT :GOTO
  310
250 PRINT :PRINT "Enter cipher message
  exactly as received, using
260 PRINT "space bar but no
  punctuation.  Do not try to correct
270 PRINT "errors as this will
  introduce additional errors."
280 PRINT "Touch < * > at end of
  message to initiate decode function."
290 PRINT :PRINTTAB(20) "UPPERCASE
  LETTERS ONLY!" :PRINT
300 PRINT "BEGIN:"
310 Z$=INKEY$ :IF Z$="" THEN 310
320 IF Z$="*" THEN 420
330 PRINT Z$;
340 IF Z$=" " THEN 310
350 IF ASC(Z$)=13 THEN 310
360 D=ASC(Z$)-64
370 IF C$="DE" THEN C$(I)=MID$(A$,D,1)
380 IF C$="EN" THEN C$(I)=MID$(B$,D,1)
390 I=I+1
400 IF I>500 THEN 420
410 GOTO 310
420 PRINT :PRINT :FOR X=1 TO I-1
430 PRINT C$(X);
440 IF C$="DE" THEN 460
450 IF RIGHT$(STR$(X),1)="0" OR
  RIGHT$(STR$(X),1)="5" THEN PRINT " ";
460 NEXT X
470 E$=RIGHT$(STR$(I-1),1)
480 IF E$="1" OR E$="6" THEN 560
490 IF E$="2" OR E$="7" THEN 550
500 IF E$="3" OR E$="8" THEN 540
510 IF E$="4" OR E$="9" THEN 530
520 GOTO 580
530 FOR X=1TO1 :GOTO 570
540 FOR X=1TO2 :GOTO 570
550 FOR X=1TO3 :GOTO 570
560 FOR X=1TO4 :GOTO 570
570 PRINT CHR$(RND(26)+64);: NEXT X
  :PRINT :PRINT
580 PRINT :PRINT "ENTER <HC> FOR HARD
  COPY PRINTOUT"
590 LINEINPUT "       <QT> TO QUIT ";D$
600 IF D$="HC" THEN 620
610 END
620 LPRINT :FOR X=1 TO I-1
630 LPRINT C$(X);
640 IF C$="DE" THEN 660
650 IF RIGHT$(STR$(X),1)="0" OR
  RIGHT$(STR$(X),1)="5" THEN
  LPRINT " ";
660 NEXT X
670 IF E$="1" OR E$="6" THEN 750
680 IF E$="2" OR E$="7" THEN 740
690 IF E$="3" OR E$="8" THEN 730
700 IF E$="4" OR E$="9" THEN 720
710 LPRINT :END
720 FOR X=1TO1 :GOTO 760
730 FOR X=1TO2 :GOTO 760
740 FOR X=1TO3 :GOTO 760
750 FOR X=1TO4 :GOTO 760
760 LPRINT CHR$(RND(26)+64); :NEXT X
  :LPRINT :LPRINT
770 END
```

BRKSEL

by Jim Barbarello

What could be worse than accidentally pressing the break key when running a Basic program on your Model II? Disable break using Debug.

Model II owners know that inadvertently pressing the break key halts any Basic program. Any further keyboard input can modify the program currently in the machine by deleting lines or changing variable values. Other undesirable results, such as data files stuck in the open mode, portions of the disk directory damaged, and the boot tracks on the disk damaged, are also possible.

I have a simple remedy for the break-key problem. My short machine-language program, BRKSEL, allows you to enable or disable the break key as simply as turning your living room lamp on and off. BRKSEL can be used in any Basic program.

Creating BRKSEL

Creating the BRKSEL program requires two steps. You must enter the BRKSEL program into the computer's memory, and transfer that program from memory to a disk file that you can use. These two steps are accomplished with another machine-language program (Debug) already part of the TRSDOS 2.0a operating system. Debug lets you view and change the contents of a large portion of the computer's memory. It also allows you to permanently copy some portion of the memory to a disk file. To make the procedure as simple as possible, we'll break it into steps.

- Start with the computer off. If it is not off, end any program, remove all disks, and turn the power off.
- Use any System disk (one that allows you to start up the machine) that doesn't automatically run a program after startup. Turn on the machine, entering the date and time responses as normal.
- Once in the TRSDOS ready mode, proceed to the next step.
- Type Debug On and press enter. Then type Debug and press enter. The screen will look like Fig. 1.
- Press the M key (to examine/change Memory). Next to the question-mark prompt will be M A = ... Type EFDA (do not press enter). The screen will look like Fig. 2.
- Press the F1 key. The cursor will move to the first displayed number on the EFDA line. Type in the following 38 characters exactly (do *not* press enter or the spacebar. Make sure the caps key is lit):

```
7EFE0020072100003E03CFC92169603E03CFC9
```

- Review the screen. If any character is incorrect, use the right and left-arrow keys to position the cursor over the incorrect character. Then type the correct character. (In this string the 0s are the number zero, *not* the letter O.) When all characters are correct, press the keyboard's F2 key (this places the characters into the machine's memory). See Fig. 3.
- Now press the O (not zero) key (this turns Debug off). The characters now stored in memory can be transferred to a disk file. The last message you see will be TRSDOS ready.
- Now type the following:

```
DUMP BRKSEL{START=EFDA, END=EFFF}
```

Check what you have typed. If it is not correct, use the backspace key to return the cursor to the offending position and retype the correct characters. When everything is correct, press enter. The disk drive will start, and after some time, you will again see TRSDOS ready. Type DIR and press enter. Your directory has a new entry entitled BRKSEL.

Using BRKSEL

BRKSEL can only be used by Basic programs or in the Basic command mode. The first line of your Basic program must include the following: CLEAR xxxxx,61401:SYSTEM "LOAD BRKSEL":DEFUSR=&HEFDA, where xxxxx is any number less than 32000, and is the amount of storage space you wish to reserve (if not specified, Basic performs an automatic CLEAR 50).

To disable the break key, you must perform a USR(0) anywhere in the program, or in the Basic command mode with a command such as X=USR(0) or BRKOFF=USR(0). The variable to the left of the equal sign must not be a string or contain any reserved words (such as For or On).

To enable (or re-enable) the break key, use the command USR(x), where x is any number between 1 and 255.

A Test Program

When in the Basic mode, type in the Program Listing.

List the program to make sure it's in memory, and run it. Follow the instructions and note that the break key has no effect when it has been disabled by the program, and then causes the program to end after it has been re-enabled.

When the program has ended, you are in the Basic command mode. Press the break key and note that it is functioning (you will see ^C each time you

See List of Advertisers on Page 563

press it). Now type X=USR(0) and press enter—the break key has no effect. Type X=USR(255) and press enter—the break key again performs normally. You can now use BRKSEL or transfer it to any of your other disks using the TRSDOS command Move. (See pages 120–121 of your TRSDOS 2.0a reference manual for further information on the Move command.)

```
10 CLS:CLEAR 500,61401:SYSTEM"LOAD BRKSEL":DEFUSR=&HEFDA
20 PRINT"THE BREAK KEY IS NOW FUNCTIONING."
30 FOR I=1 TO 1000:NEXT I:BRKOFF=USR(0):PRINT
40 PRINT"I HAVE DISABLED THE BREAK KEY. TRY PRESSING IT AND NOTE
   THAT THE PROGRAM IS NOT INTERRUPTED."
50 PRINT"WHEN YOU ARE CONVINCED, PRESS 'F1' TO CONTINUE..."
60 A$=INKEY$:IF A$=""THEN 60 ELSE IF ASC(A$)<>1 THEN 60
70 BRK=USR(1):PRINT
80 PRINT"THE BREAK KEY IS NOW FUNCTIONING. WHEN YOU PRESS IT THE
   PROGRAM WILL END."
90 A$=INKEY$:GOTO 90
```

Program Listing

Some BRKSEL Pointers

In BRKSEL's most simple application, execute a USR(0) immediately after the DEFUSR statement. At the end of the program, execute a USR(1) to re-enable the break key.

However, at some time you might not want the break key disabled. Say your program included a routine for disk access. You might wish to enable break before entering this routine and again disable it upon exit. This would allow you to manually regain control if some problem occurred in disk access. Other subtasks, such as a mathematical iterative process that could become an endless loop (never converge), are prime candidates for consideration.

If you get into a situation where you cannot regain control, your only recourse is to reset. Of course, this will leave any disk files stuck in the open mode. To properly close the file after restarting, follow these steps:

- Manually open the file by entering OPEN"D",1,"file name",xx where file name is the name of the file, and xx is the record length (this is for random/direct-access files). This command will cause an AO (already open) or similar error.
- Enter "close." This should close the stuck file. You can verify this by obtaining a directory and noting that there is no question mark after the file's name. For obstinate files, you may have to perform this procedure twice.
- Make sure you modify your program to allow manual override with the break key during the portion that caused you to have to reset. ■

Jim Barbarello can be reached at R.D. #1, Box 241H, Tennent Road, Englishtown, NJ 07726.

```
TRS-80 Model II DEBUG Program
2800     C3 7F 2E 5D 44 7A 44 65   43 1D 45 50 43 D8 5E 58    ...]DzDeC.EPC.^X
2810     61 F8 3A D9 4E C8 4F B3   41 2B 4F 3C 50 42 50 CF    a.:.N.O.A+O<PBP.
2820     50 E4 50 C0 61 C3 61 C6   61 33 65 86 65 99 65 A7    P.P.a.a.a3e.e.e.
2830     61 AA 61 AD 61 5D 44 91   44 BD 44 0A 45 E3 5C DB    a.a.a]D.D.D.E.\.
2840     5A E0 5A E5 5A A1 5D EF   5C FF 5C 2E 5D 47 5D 77    Z.Z.Z.].\.\.]G]w
2850     5D 80 5D C5 4E 44 C6 4F   52 C3 4C 53 D2 41 4E 44    ].].ND.OR.LS.AND
2860     4F 4D CE 45 58 54 C4 41   54 41 C9 4E 50 55 54 C4    OM.EXT.ATA.NPUT.
2870     49 4D D2 45 41 44 CC 45   54 C7 4F 54 4F D2 55 4E    IM.EAD.ET.OTO.UN
   PC     SP   SZHPNC  AF   BC   DE   HL   IX   IY   AF'  BC'  DE'  HL'
 2800   21FE  000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
? P
DEBUG is now ON
TRSDOS READY
DEBUG
```

Figure 1

```
TRS-80 Model II DEBUG Program
EFDA     00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
EFEA     00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
EFFA     00 00 00 00 00 00 02 FF   FF FF FF FF FF FF FF FF    ................
F00A     FF FF FF FF FF FF FF FF   FF FF FF FF FF FF FF FF    ................
F01A     FF FF FF FF FF FF FF FF   FF FF FF FF FF FF FF FF    ................
F02A     FF FF FF FF FF FF FF FF   FF FF FF FF FF FF FF FF    ................
F03A     FF FF FF FF FF FF 00 00   00 00 00 00 00 00 00 00    ................
F04A     00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
   PC     SP   SZHPNC  AF   BC   DE   HL   IX   IY   AF'  BC'  DE'  HL'
 2800   21FE  000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
? P
```

Figure 2

```
TRS-80 Model II DEBUG Program
EFDA     7E FE 00 20 07 21 00 00   3E 03 CF C9 21 69 60 3E    ... .!..>...!i`>
EFEA     03 CF C9 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
EFFA     00 00 00 00 00 00 02 FF   FF FF FF FF FF FF FF FF    ................
F00A     FF FF FF FF FF FF FF FF   FF FF FF FF FF FF FF FF    ................
F01A     FF FF FF FF FF FF FF FF   FF FF FF FF FF FF FF FF    ................
F02A     FF FF FF FF FF FF FF FF   FF FF FF FF FF FF FF FF    ................
F03A     FF FF FF FF FF FF 00 00   00 00 00 00 00 00 00 00    ................
F04A     00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00    ................
   PC     SP   SZHPNC  AF   BC   DE   HL   IX   IY   AF'  BC'  DE'  HL'
 2800   21FE  000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
?
```

Figure 3

See List of Advertisers on Page 563

Model II Business Bar Graphs

by Richard Harkness

The Model II is a powerful business micro, but its graphics capabilities fall a bit short. Now you can use yours for bar graphs!

One of the shortcomings of the Model II is its lack of graphics capability. The Models I and III produce effective, low-resolution graphics via a video grid consisting of 128 points horizontally and 48 lines vertically. This gives a total of 6,144 print positions. In contrast, the Model II has only the coarse graphics capability of its literal video display format of 80 characters horizontally and 24 lines vertically—a total of 1,920 print positions. Still, you can program the Model II to produce and print out simple bar graphs that look like those produced on the Models I and III.

This is done with a staple bar-graph program written in Model II Disk Basic that can be adapted for almost any use with minor modifications. You can relabel the axes to match the data. You can even use the RENUM command to renumber the lines starting at a higher number (e.g., 5000), and then Merge it for use as a subroutine to graph data produced by another program. In this example, I graphed monthly electricity expenses. Up to 24 bars can be drawn across the screen.

The program prompts you to input data for as many months as you want graphed, starting with January. After it draws the bars, it asks if you want to graph the same months for a prior year. If you answer yes, it prompts you to enter these values, and it graphs them beside the others. You get a comparative analysis of monthly expense over a two-year period—enough time to pick up trends, so you can determine in which months the use of central air conditioning is the heaviest and whether you've improved on energy conservation from one year to the next.

You can substitute data and read statements for the input statements to get the values graphed, if you want to save the data along with the program.

A practical limitation is the 24-by-80 grid. In this program you are limited to 18 gradations on the vertical scale—from zero to $180 in steps of $10. If

```
*BARS ON LEFT ARE FOR PRIOR YEAR*                      ELECTRICITY EXPENSE

    180:                                                                        : 180
    170:                                                                        : 170
    160:                                                                        : 160
    150:                                                                        : 150
    140:                               *                                        : 140
    130:                               *                                        : 130
    120:                               *                                        : 120
    110:      #                  *     *                 *    #            *    : 110
  $ 100: *    #      *          #*     *          #*     *    #      *    #*    : 100 $
     90:#*    #*     *          #*    #*    #     #*    #*    #*    #*    #*    : 90
     80:#*    #*    #*     *    #*    #*    #*    #*    #*    #*    #*    #*    : 80
     70:#*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    : 70
     60:#*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    : 60
     50:#*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    : 50
     40:#*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    : 40
     30:#*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    : 30
     20:#*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    : 20
     10:#*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    #*    : 10
     0 :========================================================================: 0
        JAN   FEB   MAR   APR   MAY   JUN   JUL   AUG   SEP   OCT   NOV   DEC
                                   MONTH
```

Fig. 1. Sample Printout of BARGRF1P

The Key Box

Model II or 16
32K RAM
Disk Basic
TRSDOS 1.1 or 2.0
One Disk Drive
Printer (optional)

COLOR COMPUTER

COMPUTERWARE® has it all!

FUN & GAMES

PAC ATTACK

Incredibly challenging graphics game with great sound and action.

cassette **$24.95**
disk **$29.95**

Lots more games & products available!

STARSHIP CHAMELEON

Defend your starship's planet against Gabalatok attack of bombs, anti-matter & aerial mines. Fast action, graphics, & sound.

cassette **$24.95**
disk **$29.95**

EL DIABLERO

You awaken dazed, in the middle of the desert. Your mentor sorcerer has diappeared. You face the evil El Diablero alone! Pure adventure!

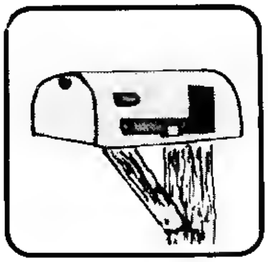
cassette **$19.95**
disk **$24.95**

HOME & WORK

HOME MONEY MANAGER

Cassette checkbook organizer with printed reports for deposits, expenses, transactions & Chart of Accounts.

$19.95



ADDRESS FACTORY

Complete name and address mailing list with special code selection and sorts for labels.

cassette **$17.95**
disk **$22.95**

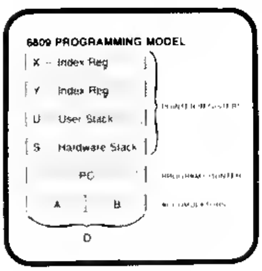
SCRIBE WORD PROCESSOR

Complete word processor (or program editor) with headings, footings, right & left justification, centering, paqination, tabs, . . . and more!

$49.95

(Cassette editor also available.)

PROGRAMMING TOOLS



MACRO ASSEMBLER

Macro conditional 6809 assembler with library files and cross reference program.

Color Computer disk **$49.95**
FLEX* disk **$50.00**

(Cassette assembler also available.)

FLEX is a trademark of TSC

PASCAL

Dynasoft's compact little PASCAL for learning structure programming. Includes compiler, P-code interpreter, editor, supervisor, & samples. (Req. 32K).

cassette **$49.95**
disk **$50.00**

Dealer Inquires Invited

TO ORDER:
Add shipping of $2 surface or $5 air/Canada. Visa & MasterCard accepted.

COMPUTERWARE®

Computerware is a trademark of Computerware. ✓369

call or write

Box 668
Encinitas, Ca. 92024
(714) 436-3512

your values go beyond these, just change the program to make the gradation larger. For example, if your top value is close to or below $1,800, use a range between zero and $1,800 in steps of $100. In a graphics representation the proportional relationships between the values are important.

The remarks in the program make most of the lines self-explanatory. Lines 330–360 define the graphics characters used to draw the scale lines, the bars, and the horizontal ruler lines. Line 410 veers to the subroutine drawing the horizontal ruler lines starting at line 1750. In line 420, EL is the ending length of each bar (line 20 on the video grid); in lines 430, 490, 560, and so on, BL is the beginning length of each bar.

The statement IF D$ = Y GOTO__ appears in several lines including line 300. It reroutes the program where needed on the second trip through (if you choose to graph months for a prior year alongside those for the current year).

You cannot obtain a printout from Program Listing 1, because it uses graphics characters unknown to most printers. Use Listing 2 if you want a printout. ■

Richard Harkness can be reached at 1224 King Henry Drive, Ocean Springs, MS 39564.

Program Listing 1

```
50 PRINT
60 PRINT
70 PRINT" THIS PROGRAM PROVIDES A GRAPHIC ANALYSIS OF YOUR MONTH
LY ELECTRICITY
80 PRINT" EXPENSE FOR A YEAR. ENTER A ZERO FOR ANY MONTH YOU WAN
T TO SKIP, BUT
90 PRINT" BE SURE YOU COUNT EACH ZERO WHEN ASKED FOR THE NUMBER
OF MONTHS YOU
100 PRINT" WANT GRAPHEO. ROUND VALUES TO NEAREST TEN DOLLARS AND
 BEGIN
110 PRINT" WITH JANUARY."
120 PRINT
130 PRINT "HOW MANY MONTHS DO YOU WANT TO GRAPH?"
140 INPUT Z
150 CLS
160 PRINT@0,"PLEASE LIST THE";Z;"VALUES TO BE GRAPHED SEPARATEO
BY A COMMA."
170 ON Z GOTO 180,190,200,210,220,230,240,250,260,270,280,290
180 INPUT A:GOTO 300
190 INPUT A,B:GOTO 300
200 INPUT A,B,C:GOTO 300
210 INPUT A,B,C,D:GOTO 300
220 INPUT A,B,C,D,E:GOTO 300
230 INPUT A,B,C,O,E,F:GOTO 300
240 INPUT A,B,C,D,E,F,G:GOTO 300
250 INPUT A,B,C,O,E,F,G,H:GOTO 300
260 INPUT A,B,C,D,E,F,G,H,I:GOTO 300
270 INPUT A,B,C,D,E,F,G,H,I,J:GOTO 300
280 INPUT A,B,C,D,E,F,G,H,I,J,K:GOTO 300
290 INPUT A,B,C,O,E,F,G,H,I,J,K,L:GOTO 300
300 IF O$="Y" GOTO 1615
310 CLS
320 PRINT CHR$(02)       'TURNS OFF BLINKING CURSOR
330 A$=CHR$(61)
340 B$=CHR$(42)
350 C$=CHR$(58)
360 L$=CHR$(144)
370 X=20
380 FOR Y=12 TO 71
390 PRINT@(X,Y),A$       'DRAWS HORIZONTAL SCALE LINE
400 NEXT Y
420 EL=19
430 BL=EL-A/10
440 IF D$="Y" GOTO 460
450 Y=13
460 FOR X=BL+1 TO EL
470 PRINT@(X,Y),B$       'DRAWS THE FIRST VERTICAL BAR
480 NEXT X
490 BL=EL-B/10
500 IF D$="Y" GOTO 1630
510 Y=18
520 FOR X=BL+1 TO EL
530 PRINT@(X,Y),B$       'DRAWS THE SECOND VERTICAL BAR
540 NEXT X
550 BL=EL-C/10
560 IF D$="Y" GOTO 1640
```

Listing continues

See List of Advertisers on Page 563

Listing 1 continued

```
570 Y=23
580 FOR X=BL+1 TO EL
590 PRINT@(X,Y),B$        'DRAWS THE THIRD VERTICAL BAR
600 NEXT X
610 BL=EL-D/10
620 IF D$="Y" GOTO 1650
630 Y=28
640 FOR X=BL+1 TO EL
650 PRINT@(X,Y),B$        'DRAWS THE FOURTH VERTICAL BAR
660 NEXT X
670 BL=EL-E/10
680 IF D$="Y" GOTO 1660
690 Y=33
700 FOR X=BL+1 TO EL
710 PRINT@(X,Y),B$        'DRAWS THE FIFTH VERTICAL BAR
720 NEXT X
730 BL=EL-F/10
740 IF D$="Y" GOTO 1670
750 Y=38
760 FOR X=BL+1 TO EL
770 PRINT@(X,Y),B$        'DRAWS THE SIXTH VERTICAL BAR
780 NEXT X
790 BL=EL-G/10
800 IF D$="Y" GOTO 1680
810 Y=43
820 FOR X=BL+1 TO EL
830 PRINT@(X,Y),B$        'DRAWS THE SEVENTH VERTICAL BAR
840 NEXT X
850 BL=EL-H/10
860 IF D$="Y" GOTO 1690
870 Y=48
880 FOR X=BL+1 TO EL
890 PRINT@(X,Y),B$        'DRAWS THE EIGHTH VERTICAL BAR
900 NEXT X
910 BL=EL-I/10
920 IF D$="Y" GOTO 1700
930 Y=53
940 FOR X=BL+1 TO EL
950 PRINT@(X,Y),B$        'DRAWS THE NINTH VERTICAL BAR
960 NEXT X
970 BL=EL-J/10
980 IF D$="Y" GOTO 1710
990 Y=58
1000 FOR X=BL+1 TO EL
1010 PRINT@(X,Y),B$       'DRAWS THE TENTH VERTICAL BAR
1020 NEXT X
1030 BL=EL-K/10
1040 IF D$="Y" GOTO 1720
1050 Y=63
1060 FOR X=BL+1 TO EL
1070 PRINT@(X,Y),B$       'DRAWS THE ELEVENTH VERTICAL BAR
1080 NEXT X
1090 BL=EL-L/10
1100 IF D$="Y" GOTO 1730
1110 Y=68
1120 FOR X=BL+1 TO EL
1130 PRINT@(X,Y),B$       'DRAWS THE TWELFTH VERTICAL BAR
1140 NEXT X
1150 PRINT@1692,"JAN"    'LABELS THE HORIZONTAL SCALE LINE
1160 PRINT@1697,"FEB"
1170 PRINT@1702,"MAR"
1180 PRINT@1707,"APR"
1190 PRINT@1712,"MAY"
1200 PRINT@1717,"JUN"
1210 PRINT@1722,"JUL"
1220 PRINT@1727,"AUG"
1230 PRINT@1732,"SEP"
1240 PRINT@1737,"OCT"
1250 PRINT@1742,"NOV"
1260 PRINT@1747,"DEC"
1270 Y=72
1280 FOR X=2 TO 20
1290 PRINT@(X,Y),B$      'DRAWS THE VERTICAL SCALE LINE
1300 NEXT X
1310 PRINT@1794,CHR$(26)" MONTH "CHR$(25)
1320 PRINT@234,"180"     'LABELS THE VERTICAL SCALE LINE
1330 PRINT@314,"170"
1340 PRINT@394,"160"
1350 PRINT@474,"150"
1360 PRINT@554,"140"
1370 PRINT@634,"130"
1380 PRINT@714,"120"
1390 PRINT@794,"110"
1400 PRINT@874,"100"
1410 PRINT@954,"90"
1420 PRINT@1034,"80"
1430 PRINT@1114,"70"
1440 PRINT@1194,"60"
1450 PRINT@1274,"50"
1460 PRINT@1354,"40"
1470 PRINT@1434,"30"
1480 PRINT@1514,"20"
1490 PRINT@1594,"10"
1500 PRINT@1674,"0"
1510 PRINT@879,"$"
1520 PRINT@50,CHR$(26)" ELECTRICITY EXPENSE "CHR$(25)
1530 IF D$="Y" GOTO 1540
1540 PRINT@0, "*BARS ON LEFT ARE FOR PRIOR YEAR*
1550 PRINT@80, "
     "
1560 IF D$= "Y" GOTO 1610
1570 PRINT@0, "GRAPH SAME MONTHS FOR PRIOR YEAR? Y=YES";
1580 INPUT D$
1590 IF D$="Y" GOTO 1600 ELSE 1610
1600 PRINT@0, "PLEASE LIST THE";Z;"VALUES FOR PRIOR YEAR
":GOTO 170
1610 GOTO 1610
1620 Y=12:GOTO 430
1630 Y=17:GOTO 520
1640 Y=22:GOTO 580
1650 Y=27:GOTO 640
1660 Y=32:GOTO 700
1670 Y=37:GOTO 760
1680 Y=42:GOTO 820
1690 Y=47:GOTO 880
1700 Y=52:GOTO 940
1710 Y=57:GOTO 1000
1720 Y=62:GOTO 1060
```

Listing 1 continues

Listing 1 continued

```
1730 Y=67:GOTO 1120
1740 GOTO 1610
1750 X=19        'BEGINS SUBROUTINE DRAWING HORIZONTAL RULER LINE
S
1760 FOR Y=12 TO 75
1770 PRINT@(X,Y),L$:NEXT Y
1780 X=18
1790 FOR Y=12 TO 75
1800 PRINT@(X,Y),L$:NEXT Y
1810 X=17
1820 FOR Y=12 TO 75
1830 PRINT@(X,Y),L$:NEXT Y
1840 X=16
1850 FOR Y=12 TO 75
1860 PRINT@(X,Y),L$:NEXT Y
1870 X=15
1880 FOR Y=12 TO 75
1890 PRINT@(X,Y),L$:NEXT Y
1900 X=14
1910 FOR Y=12 TO 75
1920 PRINT@(X,Y),L$:NEXT Y
1930 X=13
1940 FOR Y=12 TO 75
1950 PRINT@(X,Y),L$:NEXT Y
1960 X=12
1970 FOR Y=12 TO 75
1980 PRINT@(X,Y),L$:NEXT Y
1990 X=11
2000 FOR Y=12 TO 75
2010 PRINT@(X,Y),L$:NEXT Y
2020 X=10
2030 FOR Y=12 TO 75
2040 PRINT@(X,Y),L$:NEXT Y
2050 X=9
2060 FOR Y=12 TO 75
2070 PRINT@(X,Y),L$:NEXT Y
2080 X=8
2090 FOR Y=12 TO 75
2100 PRINT@(X,Y),L$:NEXT Y
2110 X=7
2120 FOR Y=12 TO 75
2130 PRINT@(X,Y),L$:NEXT Y
2140 X=6
2150 FOR Y=12 TO 75
2160 PRINT@(X,Y),L$:NEXT Y
2170 X=5
2180 FOR Y = 12 TO 75
2190 PRINT@(X,Y),L$:NEXT Y
2200 X=4
2210 FOR Y=12 TO 75
2220 PRINT@(X,Y),L$:NEXT Y
2230 X=3
2240 FOR Y=12 TO 75
2250 PRINT@(X,Y),L$:NEXT Y
2260 X=2
2270 FOR Y=12 TO 75
2280 PRINT@(X,Y),L$:NEXT Y
2290 RETURN
```

Program Listing 2

```
50 PRINT
60 PRINT
70 PRINT" THIS PROGRAM PROVIDES A GRAPHIC ANALYSIS OF YOUR MONTH
LY ELECTRICITY
80 PRINT" EXPENSE FOR A YEAR. ENTER A ZERO FOR ANY MONTH YOU WAN
T TO SKIP, BUT
90 PRINT" BE SURE YOU COUNT EACH ZERO WHEN ASKED FOR THE NUMBER
OF MONTHS YOU
100 PRINT" WANT GRAPHED. ROUND VALUES TO NEAREST TEN DOLLARS AND
 BEGIN
110 PRINT" WITH JANUARY."
120 PRINT
130 PRINT "HOW MANY MONTHS DO YOU WANT TO GRAPH?"
140 INPUT Z
150 CLS
160 PRINT@0,"PLEASE LIST THE";Z;"VALUES TO BE GRAPHED SEPARATED
BY A COMMA."
170 ON Z GOTO 180,190,200,210,220,230,240,250,260,270,280,290
180 INPUT A:GOTO 300
190 INPUT A,B:GOTO 300
200 INPUT A,B,C:GOTO 300
210 INPUT A,B,C,D:GOTO 300
220 INPUT A,B,C,D,E:GOTO 300
230 INPUT A,B,C,D,E,F:GOTO 300
240 INPUT A,B,C,D,E,F,G:GOTO 300
250 INPUT A,B,C,D,E,F,G,H:GOTO 300
260 INPUT A,B,C,D,E,F,G,H,I:GOTO 300
270 INPUT A,B,C,D,E,F,G,H,I,J:GOTO 300
280 INPUT A,B,C,D,E,F,G,H,I,J,K:GOTO 300
290 INPUT A,B,C,D,E,F,G,H,I,J,K,L:GOTO 300
300 IF D$="Y" GOTO 1620
310 CLS
320 PRINT CHR$(02)        'TURNS OFF BLINKING CURSOR
330 A$=CHR$(152)
340 B$=CHR$(158)
350 C$=CHR$(151)
360 L$=CHR$(144)
370 X=20
380 FOR Y=12 TO 71
390 PRINT@(X,Y),A$        'DRAWS HORIZONTAL SCALE LINE
400 NEXT Y
410 GOSUB 1750
420 EL=20
430 BL=EL-A/10
440 IF D$="Y" GOTO 460
450 Y=13
460 FOR X=BL+1 TO EL
470 PRINT@(X,Y),B$        'DRAWS THE FIRST VERTICAL BAR
480 NEXT X
490 BL=EL-B/10
500 IF D$="Y" GOTO 1630
510 Y=18
520 FOR X=BL+1 TO EL
530 PRINT@(X,Y),B$        'DRAWS THE SECOND VERTICAL BAR
540 NEXT X
550 BL=EL-C/10
560 IF D$="Y" GOTO 1640
```

Listing 2 continues

Listing 2 continued

```
570 Y=23
580 FOR X=BL+1 TO EL
590 PRINT@(X,Y),B$        'DRAWS THE THIRD VERTICAL BAR
610 BL=EL-D/10
620 IF D$="Y" GOTO 1650
630 Y=28
640 FOR X=BL+1 TO EL
650 PRINT@(X,Y),B$        'DRAWS THE FOURTH VERTICAL BAR
660 NEXT X
670 BL=EL-E/10
680 IF D$="Y" GOTO 1660
690 Y=33
700 FOR X=BL+1 TO EL
710 PRINT@(X,Y),B$        'DRAWS THE FIFTH VERTICAL BAR
720 NEXT X
730 BL=EL-F/10
740 IF D$="Y" GOTO 1670
750 Y=38
760 FOR X=BL+1 TO EL
770 PRINT@(X,Y),B$        'DRAWS THE SIXTH VERTICAL BAR
780 NEXT X
790 BL=EL-G/10
800 IF D$="Y" GOTO 1680
810 Y=43
820 FOR X=BL+1 TO EL
830 PRINT@(X,Y),B$        'DRAWS THE SEVENTH VERTICAL BAR
840 NEXT X
850 BL=EL-H/10
860 IF D$="Y" GOTO 1690
870 Y=48
880 FOR X=BL+1 TO EL
890 PRINT@(X,Y),B$        'DRAWS THE EIGHTH VERTICAL BAR
900 NEXT X
910 BL=EL-I/10
920 IF D$="Y" GOTO 1700
930 Y=53
940 FOR X=BL+1 TO EL
950 PRINT@(X,Y),B$        'DRAWS THE NINTH VERTICAL BAR
960 NEXT X
970 BL=EL-J/10
980 IF D$="Y" GOTO 1710
990 Y=58
1000 FOR X=BL+1 TO EL
1010 PRINT@(X,Y),B$       'DRAWS THE TENTH VERTICAL BAR
1020 NEXT X
1030 BL=EL-K/10
1040 IF D$="Y" GOTO 1720
1050 Y=63
1060 FOR X=BL+1 TO EL
1070 PRINT@(X,Y),B$       'DRAWS THE ELEVENTH VERTICAL BAR
1080 NEXT X
1090 BL=EL-L/10
1100 IF D$="Y" GOTO 1730
1110 Y=68
1120 FOR X=BL+1 TO EL
1130 PRINT@(X,Y),B$       'DRAWS THE TWELFTH VERTICAL BAR
1140 NEXT X
1150 PRINT@1692,"JAN"    'LABELS THE HORIZONTAL SCALE LINE
1160 PRINT@1697,"FEB"
1170 PRINT@1702,"MAR"
1180 PRINT@1707,"APR"
1190 PRINT@1712,"MAY"
1200 PRINT@1717,"JUN"
1210 PRINT@1722,"JUL"
1220 PRINT@1727,"AUG"
1230 PRINT@1732,"SEP"
1240 PRINT@1737,"OCT"
1250 PRINT@1742,"NOV"
1260 PRINT@1747,"DEC"
1270 Y=72
1280 FOR X=2 TO 20
1290 PRINT@(X,Y),C$      'DRAWS THE VERTICAL SCALE LINE
1300 NEXT X
1302 Y=11
1304 FOR X=2 TO 20
1306 PRINT@(X,Y), C$
1308 NEXT X
1310 PRINT@1794,CHR$(26)" MONTH "CHR$(25)
1320 PRINT@234,"180":PRINT@168,"180"    'LABELS VERTICAL SCALE L
INES
1330 PRINT@314,"170":PRINT@248,"170
1340 PRINT@394,"160":PRINT@328,"160
1350 PRINT@474,"150":PRINT@408,"150
1360 PRINT@554,"140":PRINT@488,"140
1370 PRINT@634,"130":PRINT@568,"130
1380 PRINT@714,"120":PRINT@648,"120
1390 PRINT@794,"110":PRINT@728,"110
1400 PRINT@874,"100":PRINT@808,"100
1410 PRINT@954,"90":PRINT@889,"90
1420 PRINT@1034,"80":PRINT@969,"80
1430 PRINT@1114,"70":PRINT@1049,"70
1440 PRINT@1194,"60":PRINT@1129,"60
1450 PRINT@1274,"50":PRINT@1209,"50
1460 PRINT@1354,"40":PRINT@1289,"40
1470 PRINT@1434,"30":PRINT@1369,"30
1480 PRINT@1514,"20":PRINT@1449,"20
1490 PRINT@1594,"10":PRINT@1529,"10
1500 PRINT@1674,"0":PRINT@1609,"0
1510 PRINT@878,"$":PRINT@806,"$"
1520 PRINT@50,CHR$(26)" ELECTRICITY EXPENSE "CHR$(25)
1530 IF D$="Y" GOTO 1540 ELSE 1570
1540 PRINT@0, "*BARS ON LEFT ARE FOR PRIOR YEAR*                    "
1550 PRINT@80, "
     "
1555 IF E$="Y" GOTO 1870
1560 IF D$= "Y" GOTO 1840
1570 PRINT@0, "GRAPH SAME MONTHS FOR PRIOR YEAR?";
1575 PRINT@80, "CHOOSE ONE: Y=YES; N=N0";
1580 INPUT D$
1585 PRINT@80, "                                   "
1590 IF D$="Y" GOTO 1600
1592 PRINT@0, "                                                "
1594 GOTO 1840
1600 PRINT@0, "PLEASE LIST THE";Z;"VALUES FOR PRIOR YEAR
":GOTO 170
```

Listing 2 continues

Listing 2 continued

```
1610 GOTO 1610
1615 BS=CHR$(35)
1620 Y=12:GOTO 430
1630 Y=17:GOTO 520
1640 Y=22:GOTO 580
1650 Y=27:GOTO 640
1660 Y=32:GOTO 700
1670 Y=37:GOTO 760
1680 Y=42:GOTO 820
1690 Y=47:GOTO 880
1700 Y=52:GOTO 940
1710 Y=57:GOTO 1000
1720 Y=62:GOTO 1060
1730 Y=67:GOTO 1120
1740 GOTO 1610
1840 PRINT@80, "DO YOU WANT A PRINTOUT? Y=YES";
1850 INPUT E$
1860 IF E$="Y" GOTO 1870 ELSE 1610
1870 PRINT@80, "PRINTER READY? Y=YES
1880 INPUT F$
1890 IF F$="Y" GOTO 1910 ELSE 1900
1900 PRINT@160," ":GOTO 1840
1910 PRINT@80, "
"
1920 PRINT@160, "
1930 SYSTEM "SCREEN"
1940 GOTO 1690
```


✓See List of Advertisers on Page 563

Mod II Disk Index

by Charles R. Perelman

Do you find it hard to keep track of all your disk files, and even harder to read the directories? This disk index utility is for you!

When was the last time you shuffled through your disks or card index to find a program for your Model II and muttered that there must be a better way? Here's a disk index utility that reads each disk and keeps track of your programs.

A few commands used are unique to Model II Basic, but can be readily translated into other TRS-80 dialects. Machine-language USR routines are included that take advantage of Model II supervisory calls; the program also uses a machine-language sort and PEEK in a USR routine. You do not need to know anything about machine language to use the program.

Simply put the program disk in drive 0 and insert each disk to be indexed in your second disk drive. The computer reads the directory and processes the information.

You must give your disks identifying numbers. For easy handling, I put sequential self-adhesive numbers purchased at a stationery store on each disk.

Main menu options permit you to create a new index, update, add a disk, delete a disk, list the index alphabetically or in disk sequence, list the contents of a specific disk, list all disks on which a particular program can be found, or save the revised index to disk.

The basic organizational structure and the vertical raster routine were borrowed from "Lost and Found," a directory program for the Model III by

Program Listing

```
0 '****************************************
10 'DIRECTORY PROGRAM FOR TRS-80 MODEL II
20 'ADAPTED FROM PROGRAM BY ATHANASIOU 80 MICRO, JUNE 82
30 'BY CHARLES R. PERLEMAN, A PROFESSIONAL CORPORATION
40 '9777 WILSHIRE BLVD., BEVERLY HILLS, CA. 90212
50 'PROGRAM FILE NAME 'DIRECTRY/MII'
100 '****************************************
110 'SET PARAMETERS
120 CLEAR 20000,62270
130 DEFINT A,J-M,U-Y
140 DEFSTR F,P,D,C
150 DIM PN$(600),DP$(600),U1(23),A(96),U5(93)
160 DIM X,Y,XX,J,I:J=0:V9=0
170 S$(1)="   TO RESTART TYPE:  'BASIC DIRECTRY/MII -F:1' AND PR
ESS <ENTER>.":S$(2)="*** WORKING...PLEASE WAIT ***":S$(3)="PRESS
 ENTER TO CONTINUE":S$(4)="PRESS <M> FOR MENU"
200 '*************************************
210 'IMPLEMENT PEEK AND SET UP REGISTERS FOR USR ROUTINES
220 FOR X=0 TO 23:READ U1(X):NEXT
230 V=0:DEFUSR=VARPTR(U1(0)):V=USR(0)
240 U2(0)=15872:U2(2)=8448:U2(6)=256:U2(8)=-13873
250 FOR X=0 TO 93:READ U5(X):NEXT
260 ON ERROR GOTO 11020
300 '************************************
310 'FILES TO BE OPENED FOR MODEL II
320 CLS:PRINT"MODEL II REQUIRES THAT FILES BE OPENED WHEN ENTERI
NG BASIC.":PRINT"IF YOU NEGLECTED TO DO THIS, PRESS <BREAK>, ENT
ER 'SYSTEM' TO":PRINT"RETURN TO TRSDOS.":PRINT S$(1):PRINT:PRINT
```

Listing continues

The Key Box

Model II or 16
64K RAM
Disk Basic
TRSDOS 1.2 or 2.0, 2 disk drives

SUPER PRICES & FREE SHIPPING

NEW LOW PRICES

COMPUTERS

Complete MODEL III COMPUTER
Model III with 48K memory, 2 disk drives, 370K storage, and ready to run with TRSDOS 1.3 and manual. 120 day Simutek warranty parts and labor. Compatible with all Radio Shack software $1799.99

Model III with 48K, 2 double sided 40 track disk drives with 750K storage. Comes with MULTIDOS and manual. 120 day Simutek warranty parts and labor. Compatible with Radio Shack software. Complete and ready to run $2195.99

MODEL III INTERNAL DISK DRIVE KITS
If you can use a phillips screwdriver, you can easily install Simutek's Model III disk drives and controller in less than an hour. Absolutely NO SOLDERING, TRACE CUTTING OR TECHNICAL KNOWLEDGE REQUIRED! The J & M controller is the best on the market today. We've tried others and found J & M's the finest and easiest to install. We warranty the J & M controller for 120 days against defects in workmanship.

ONE 40 TRACK TANDON DISK DRIVE WITH 185K STORAGE, and J & M controller. Radio Shack DOS, manual. No soldering or trace cutting required. Ready to install with instructions $589.99

TWO 40 TRACK TANDON DISK DRIVES 370K STORAGE, with J & M controller Radio Shack DOS, manual. No soldering or trace cutting. Ready to install and run. Instructions $824.99

TWO DOUBLE SIDED 40 TRACK DISK DRIVES with 750K storage and J & M controller MULTIDOS and manual and Model III DOS. No soldering or trace cutting. Ready to install and run. Full instructions $999.99

MODEL III HARD DISK DRIVES
FIVE MEGABYTE hard disk with LDOS disk operating system, all cables, connectors, ready to plug in $1799.99

TEN MEGABYTE Hard disk with LDOS, all cables, connectors ready to plug in $1999.99

PRINTERS

DAISY WHEEL PRINTERS:	
NEW SMITH CORONA DAISY WHEEL 12 CPS	$699.99
NEW BROTHER DAISY WHEEL 16 CPS	$1199.95
TRACTOR FEED FOR BROTHER	$250.00
C-ITOH STARWRITER II F 10. 40 CPS	$1649.99
C-ITOH STARWRITER III F-10 55CPS	$1999.99
TRACTOR FEED FOR STARWRITERS	$250.00
HIGH QUALITY DOT MATRIX:	
C-ITOH PROWRITER I 8510A .2K BUFF...PRL. ONLY	$499.99
C-ITOH PROWRITER I 8510ACD...2K BUFF. SER/PRL	$649.99
C-ITOH PROWRITER II 1550...15 INCH...3K BUFFER	$799.99
EPSON MX-80 WITH GRAFTRAX	$479.99
EPSON MX-80/FT WITH GRAFTRAX	$549.99
EPSON MX-100	$799.99
OKIDATA MICROLINE 80. (80 CPS)	$369.99
OKIDATA MICROLINE 82A (125 CPS) W/TRACTOR	$499.99
OKIDATA MICROLINE 83A (125 CPS) W/TRACTOR	$749.99
OKIDATA MICROLINE 84 (200 CPS) W/TRACTOR	$1199.95
PRINTER CABLE FOR MOD I EXPANSION INTERFACE OR MODEL III (SPECIFY)	$29.95
PROWRITER RIBBONS	$12.99
EPSON MX-80 OR FT RIBBONS	$9.99
OKIDATA RIBBONS...(80, 82A, 83A)	$3.49

DISK DRIVES

MODEL I DISK DRIVES
ONE TEC DISK DRIVE 40 TRACK DISK DRIVE with 5ms track to track access time, with power supply chassis, extender cable and shipping for TRS-80. One year warranty on parts and labor. Ten day money back guarantee.
Model I or III $279.00

TWO 40 TRACK TEC DISK DRIVES, with power supply and cable
One year warranty $539.95

ONE TANDON 40 TRACK DISK DRIVE, 5 ms track to track, with P/S and chassis, ready to run as drive 0-4 or Model III external. 10 day money back warranty $279.00

ONE 2 DRIVE cable for Model I or III. (Specify) $25.00

ONE 4 DRIVE cable for Model I only $35.00

ACCESSORIES/ SOFTWARE, ETC.

SALE!
DOSPLUS 3.4
MOD I ONLY
$69.95
While supply lasts.

16K MEMORY UPGRADE FOR MOD I, III OR COLOR 4K. UNCONDITIONALLY GUARANTEED FOR 2 YEARS	$14.95
DISK OPERATING SYSTEMS	
NEWDOS 80 VER. 2	$139.95
LDOS	$129.95
DOSPLUS	$139.99
MULTIDOS. BEST BUY! SPECIFY DOUBLE OR SINGLE DENS	$79.95
DOUBLE DENSITY MOD I	
PERCOM DOUBLER II. (NEW LOW PRICE!)	$139.99
With DOSPLUS 3.4	$169.95
DISKETTE STORAGE	
SMOKED PLASTIC HOLDS 50 DISKETTES	$21.95
HARD BOX HOLD TEN	$3.99

MODEMS	
LYNX MODEM FOR MOD I OR III	$239.95
OMNITERM SMART TERMINAL PROGRAM	$99.99
SUPER BOOKS FROM IJG	
MICROSOFT BASIC DECODED AND OTHER MYSTERIES	$29.95
TRS-80 DISK AND OTHER MYSTERIES	$22.50
BASIC FASTER AND BETTER AND OTHER MYSTERIES	$29.95
BASIC FASTER AND BETTER PROGRAMS ON DISKETTE	$19.95
CUSTOM TRS-80 AND OTHER MYSTERIES	$29.95
MONITORS	
OLD MOD I SCREEN STARTING TO GIVE YOU HEADACHES? TRY ONE OF THESE HIGH RESOLUTION MONITORS AT SUPER LOW PRICES...	
ZENITH GREEN SCREEN 12"	$139.95
BMC 12 INCH GREEN SCREEN	$109.95
BMC 13 INCH COLOR	$299.99
AMDEK 12" BLACK AND WHITE	$97.50
OCI AMBER SCREEN MONITOR 12"	$189.95
CABLE TO CONNECT MONITORS TO MODEL I COMPUTER	$14.95

C.O.D.'S $5.00 EXTRA. NO C.O.D.'S FOR MODEL III COMPUTERS OR PARCELS THAT CANNOT BE SHIPPED UNITED PARCEL SERVICE. ALL C.O.D.'S REQUIRE CASH OR CERTIFIED CHECK AT TIME OF COLLECTION.

CALL TOLL FREE to order:

1-800-528-1149

WE ACCEPT AMERICAN EXPRESS, VISA and MASTERCARD.
NO SURCHARGE FOR CREDIT CARD ORDERS!
SORRY, PERSONAL CHECKS REQUIRE 4-5 WEEKS TO CLEAR.
MINIMUM ORDER $20.00

~164

TRS-80 TM of Radio Shack a Tandy corp.

SIMUTEK
Computer Products Inc.
4897 E. SPEEDWAY BLVD., TUCSON, AZ 85712, (602) 323-9391

Robert and William Athanasiou (*80 Micro*, June/July 1982, page 288). If you are using that program, you can enhance it, yet reduce the size of many of the program routines by adapting my coding. I have substituted INKEY$ routines for most of the Input entries, eliminated duplication, and made other structural changes.

Construction of the program is modular. Each section performs a separate function. You can easily debug each part, make changes to it, or use a module in other programs. The remark lines are not referenced in the program and you can delete them in your final work copy to increase available memory. You can remove spaces for further compression.

The dividing remark lines are numbered with even thousands. Descriptive lines end with the digits 10. All program lines are divisible by 10.

Program Overview

The entire directory of the disk in drive 1 is read by a machine-language USR routine. The computer extracts the name of each program and adds the disk identification number for indexing. Except for creation of a new index, your prior index is read automatically from disk before the program proceeds. To update a disk, delete the prior disk information, then read the present directory into the index.

As you review the contents of the index, the headings do not scroll off the screen, but remain in place. Up to 80 entries are on screen at one time. You can page back and forth from the beginning to the end of your index. The entire index may be printed in alphabetical or disk number sequence.

There are no page breaks in the printer routine. You can either stop the printer with the hold key to load another sheet of paper or use continuous roll or fanfold. The format for printing is 80 characters wide so that you can use standard 8 1/2-inch paper.

A flashing error message alerts you if there is no prior index without terminating the program. If you request a printout without turning on the printer, an error message prompts you so you can try again. At the beginning of the program, you are reminded to open a file as required by Model II TRSDOS. Should you forget to do this, you will be unable to save a revised index to disk. If you try to access a disk file without having opened a file, you will be returned to TRSDOS to start again.

Detailed Tutorial

The program contains multiple Basic statements on a single line separated by colons. This runs faster than individual lines. The If...Then...Else format reduces the number of lines.

- Line 120 clears the maximum space to speed program operation. Memory above 62270 is reserved for the disk-read USR routine.
- Line 130 defines numerical variables as integers, reducing processing time and memory usage.
- Line 150 provides variable space for 600 programs. This is more than sufficient for my needs since many of my programs are CP/M format.
- Line 170 avoids retyping repeated statements by defining them as strings.
- In line 220, hexadecimal coding reads from the data statement at the end of the program for the PEEK function.
- Line 230's USR routine implements the hexadecimal code to replace OCT$ in Basic with PEEK, only for this program.
- Line 240 has the coding for USR routines to prepare registers, and is used with supervisory calls to read the disk directories and stop the screen from scrolling.
- Line 250 reads a machine-language sort into an array for later call by the USR routine.
- Line 260 initializes an error trap. Any error sends the program to line 11020. When you are typing in the program, disable this line. Put a remark at the beginning. The built-in TRSDOS error routines are then available to you in debugging. After the program is completely debugged, remove the remark.
- Line 320 reminds you that files must be opened with the Model II when you bring Basic into memory.
- Line 330 pauses, holding information on the screen until any key is pressed.
- Line 530 names the index file produced DSKINDEX. The program recognizes only capital letters. Pressing the letter M returns you to the main menu from various points not always specifically prompted.
- Line 620 provides error traps that cause a flashing message if your choice is beyond the limits given, or is not a whole number.
- Line 630 sets flag V2 to 1, if your choice is between 1 and 4. Later, this flag reminds you that you must save the file to disk to retain index changes.
- In line 640, a K value from 1 to 8 directs the computer to the appropriate program section.
- In line 1020, after a subroutine to obtain the disk identification number, each program is checked for a match. An error message flashes if the ID number is already used in the index.
- Line 1030 instructs you to insert the disk to be read and to press enter.
- In line 1040, if you press M instead of enter, the program aborts to the main menu. Otherwise, a message is printed on the screen while the disk directory is read and processed.
- Line 1050 lets you process another disk, save the file, or return to the menu by pressing a single key. The semicolon keeps the cursor immediately after the prompt rather than moving it to the next line. INSTR directs the program to the first line number after the GOTO if M is pressed, the second for D, the third for S. An error message flashes and you get another choice if the character pressed is not listed. This same type of coding is repeated several times in the program.
- In line 2020, if you have not read a disk into memory to begin a new index or recalled the prior index, DSKINDEX is read to memory from disk.
- Line 3040 compares disk numbers for all programs to the disk number you want to delete or list. The index number for each matching program is put into

Listing continued

```
 S$(3);
330 IF INKEY$="" THEN 33Ø
5ØØ '************************************
51Ø 'MAIN MENU
52Ø CLS:PRINT@165,"*** INDEX FOR MODEL II ***":PRINT
53Ø PRINT,"1.  MAKE NEW INDEX":PRINT,"2.  ADD DISK TO INDEX":PRI
NT,"3.  UPDATE DISK IN INDEX":PRINT,"4.  DELETE DISK FROM INDEX"
:PRINT,"5.  LIST ALPHABETICALLY OR BY DISK NUMBER.":PRINT,"6.  F
IND PROGRAM OR LIST A DISK."
54Ø PRINT,"7.  SAVE INDEX AS 'DSKINDEX'":PRINT,"8.  END PROGRAM"
:PRINT:PRINT"      NOTE:  USE CAPITAL LETTERS ONLY.  PRESS <CAPS
> IF NECESSARY.":PRINTTAB(13)"WHEN CHOICE IS REQUESTED AT ANY ST
AGE OF PROGRAM,":PRINTTAB(13) S$(4)
55Ø PRINT:PRINT"  PRESS NUMBER DESIRED";
6ØØ '************************************
61Ø 'INKEY INPUT FOR CHOICES
62Ø GOSUB 1ØØ2Ø:K=VAL(C$):IF K<1 OR K>8 OR K<>INT(K) THEN M=1:GO
SUB 1Ø12Ø:GOTO 52Ø
63Ø IF K<5 THEN V2=1
64Ø CLS:ON K GOTO 1Ø2Ø,2Ø2Ø,3Ø2Ø,3Ø2Ø,5Ø2Ø,6Ø2Ø,7Ø2Ø,8Ø2Ø
1ØØØ '************************************
1Ø1Ø 'PRODUCE NEW INDEX
1Ø2Ø GOSUB 1Ø32Ø:FOR X=1 TO XX:IF DN$=LEFT$(DP$(X),2) THEN M=7:G
OSUB 1Ø12Ø:GOTO 52Ø ELSE NEXT
1Ø3Ø PRINT:PRINT"INSERT DISK ";DN$;" IN DRIVE 1":PRINT:PRINT S$(
3);:GOSUB 1ØØ2Ø
1Ø4Ø IF C$="M" THEN 52Ø ELSE CLS:PRINT@658,S$(2);:DZ$=DN$:IF K>1
 THEN DY$=DP$(1)
1Ø5Ø GOSUB 1Ø42Ø:V3=1:IF K>1 THEN DP$(1)=DY$
1Ø6Ø CLS:C$="":PRINT:PRINT S$(4);" <D> TO INDEX ANOTHER DISK OR
<S> TO SAVE FILE.";:GOSUB 1ØØ2Ø:ON INSTR("MDS",C$)GOTO 52Ø,1Ø2Ø,
7Ø2Ø:M=2:GOSUB 1Ø12Ø:CLS:GOTO 1Ø6Ø
2ØØØ '************************************
2Ø1Ø 'ADD NEW DISK #
2Ø2Ø IF V3=Ø THEN GOSUB 1Ø92Ø
2Ø3Ø GOTO 1Ø2Ø
3ØØØ '************************************
3Ø1Ø 'DELETE FILE
3Ø2Ø IF V3=Ø THEN GOSUB 1Ø92Ø
```

Listing continues

the A array.

- Lines 3050–3080 sort the array and list programs in alphabetical order in horizontal sequence. The comma in 3080 causes a tab after each program name for a five-column listing. If no program matches the input disk number, an error message flashes.
- In line 3090, the same routine is used for deleting, updating, or listing the programs on a disk. K, the number of your original choice from the menu, directs the computer to the proper continuation routine.
- In lines 5030–5060, flag V4 signals an alphabetical or disk number. For alphabetical sorting, the disk number is changed from the first two characters to the last two characters of the string. Alphabetical listing, therefore, takes longer than by disk number.
- Line 5120 selects the proper heading for listing on the screen, based on the V4 flag.
- Line 5130's USR 3 calls supervisory routine 27, preventing the top two header lines from scrolling.
- Lines 5140–5150 list the index on the screen after it is processed by the vertical raster subroutine.
- In lines 5160–5200, W4 starts a new line after each four entries. The pro-

Listing continued

```
3030 GOSUB 10320:PRINT:PRINT S$(2);
3040 U=1:FOR X=1 TO XX:IF LEFT$(DP$(X),2)=DN$ THEN A(U)=X:U=U+1
3050 NEXT:IF U=1 THEN M=8:GOSUB 10120:GOTO 3030
3060 GOSUB 10530
3070 CLS:PRINT:PRINT"PROGRAMS ON DISK ";DN$;":":PRINT
3080 FOR X=1 TO U-1:PRINT RIGHT$(DP$(A(X)),12),:NEXT:PRINT:C$=""
3090 IF K>4 THEN PRINT:PRINT S$(3);:GOSUB 10020:GOTO 6030
3100 PRINT:PRINT S$(4);" OR <D> TO DELETE";:GOSUB 10020:ON INSTR
("MD",C$)GOTO 520,3110:M=2:GOSUB 10120:CLS:GOTO 3090
3110 PRINT:PRINT S$(2);:GOSUB 10720:IF K=4 THEN 520 ELSE 1030
5000 '************************************
5010 'SORT
5020 IF V3=0 THEN GOSUB 10920
5030 PRINT:PRINT"PRESS <D> FOR LISTING BY DISK, <A> FOR ALPHABET
ICAL";:GOSUB 10020:ON INSTR("MDA",C$)GOTO 520,5040,5050:M=2:GOSU
B 10120:CLS:GOTO 5030
5040 V4=0:GOTO 5060
5050 V4=1:PRINT:PRINT S$(2):FOR X=1 TO XX:DP$(X)=RIGHT$(DP$(X),1
2)+LEFT$(DP$(X),2):NEXT
5060 CLS:PRINT:PRINT:PRINT S$(2);:GOSUB 10520:CLS
5100 '****************************
5110 'SCREEN LIST
5120 FOR X=1 TO 4:IF V4=0 THEN PRINT"DISK  PROGRAM      ";:NEXT
ELSE PRINT"PROGRAM    DISK    ";:NEXT
5130 PRINT:PRINT STRING$(75,"-"):V6=0:U2(1)=27:U2(7)=512:DEFUSR3
=VARPTR(U2(0)):V6=USR3(0)
5140 XX=XX-1:FOR I=1 TO XX:GOSUB 10820
5150 IF V4=0 THEN PRINT LEFT$(DP$(J),2);"   ";RIGHT$(DP$(J),12);
"  ";ELSE PRINT LEFT$(DP$(J),12);" ";RIGHT$(DP$(J),2);"     ";
5160 IF W4=4 THEN PRINT
5170 IF INT(I/80)<>I/80 THEN 5190 ELSE PRINT:PRINT"PRESS <C> TO
CONTINUE DISPLAY <B> FOR PRIOR PAGE";:GOSUB 10020
5180 IF I>=160 AND C$="B" THEN I=I-160:GOTO 5210 ELSE 5210
5190 IF I>80 AND I=XX THEN PRINT:PRINT:PRINT"FOR PRIOR PAGE PRES
S <B>, <C> TO END LIST      ";ELSE 5220
5200 GOSUB 10020:IF C$="B" THEN I=I-(80+(XXMOD80))
5210 PRINT STRING$(48,8);
5220 NEXT:XX=XX+1:C$=""
5300 '*********************************
5310 'PRINTER OUTPUT
5320 PRINT:PRINT S$(4);", <P> FOR PRINTOUT";:GOSUB 10020:O
N INSTR("PM",C$)GOTO 5340,5330:M=2:GOSUB 10120:CLS:GOTO 5320
5330 IF V4=1 THEN PRINT:PRINT S$(2):FOR X=1 TO XX:DP$(X)=RIGHT$(
DP$(X),2)+LEFT$(DP$(X),12):NEXT:GOTO 520 ELSE 520
5340 LPRINT:FOR I=1 TO 4:IF V4=0 THEN LPRINT"DISK PROGRAM       "
;:NEXT ELSE LPRINT"PROGRAM     DISK    ";:NEXT
5350 LPRINT:LPRINT STRING$(75,"-"):W4=0:XX=XX-1:FOR I=1 TO XX:GO
SUB 10820
5360 IF V4=0 THEN LPRINT LEFT$(DP$(J),2);" ";RIGHT$(DP$(J),12);"
   ";ELSE LPRINT LEFT$(DP$(J),12);" ";RIGHT$(DP$(J),2);"    ";
5370 IF W4=4 THEN LPRINT
5380 NEXT:XX=XX+1:IF V4=1 THEN PRINT:PRINT S$(2):FOR X=1 TO XX:D
P$(X)=RIGHT$(DP$(X),2)+LEFT$(DP$(X),12):NEXT:GOTO 520 ELSE GOTO
520
6000 '************************************
6010 'SEARCH FOR DISK OR PROGRAM
6020 IF V3=0 THEN GOSUB 10920
6030 CLS:C$="":PRINT@240,S$(4);", <D> TO LIST A DISK, OR <P> TO
FIND A PROGRAM.";:GOSUB 10020:ON INSTR("MDP",C$)GOTO 520,3020,60
40:M=2:GOSUB 10120:CLS:GOTO 6030
6040 CLS:PRINT:LINEINPUT"ENTER NAME OF PROGRAM TO BE FOUND:  ";P
N$:IF PN$="M" THEN 520 ELSE J=0:L=LEN(PN$)
6050 IF L<12 THEN PN$=PN$+SPACE$(12-L) ELSE IF L>12 THEN PN$=LEF
T$(PN$,12)
6060 FOR X=1 TO XX-1:IF PN$=RIGHT$(DP$(X),12) THEN J=J+1:PN$(J)=
DP$(X)
6070 NEXT
6080 IF J=0 THEN M=8:GOSUB 10120:CLS:GOTO 6040
6090 CLS:PRINTTAB(5);"PROGRAM";TAB(20);"DISK":PRINT STRING$(55,"
-"):PRINT:FOR I=1 TO J:PRINTTAB(5);RIGHT$(PN$(I),12);" ";LEFT$(P
N$(I),2):NEXT
6100 C$="":PRINT:PRINT S$(4);" OR <P> TO FIND ANOTHER PROGRAM";:
GOSUB 10020:ON INSTR("MP",C$)GOTO 520,6040:M=2:GOSUB 10120:CLS:G
OTO 6100
7000 '************************************
7010 'SAVE TO DISK
7020 IF V3=0 THEN M=3:GOSUB 10120:GOTO 520 ELSE OPEN "O",1,"DSKI
NDEX":PRINT@658,"*** SAVING INDEX TO DISK ***";
7030 PRINT#1,XX:FOR X=1 TO XX:PRINT#1,DP$(X):NEXT:CLOSE:V3=0:M=5
:GOSUB 10120:GOTO 520
8000 '************************************
8010 'END OF PROGRAM
8020 IF V2=0 THEN 8040 ELSE PRINT"TO RETAIN YOUR UPDATED INDEX,
YOU MUST SAVE IT TO DISK.":PRINT"PRESS <S> TO SAVE OR <E> TO END
 WITHOUT SAVING.";
```

Listing continues

gram pauses with 80 items on the screen. You may go back to a prior screen. A partial final screen is controlled by lines 5190 and 5200. The MOD operator is a remainder function. XX MOD 80 gives the remainder of XX (total programs) divided by 80. This is the number of programs after the last full screen.

- Line 5210 erases the prompt message.
- In line 5220, For...Next loops always add one more than the number of items processed. Proper program operation requires adjustment at times during the program to the actual number of programs, or an increase by one as though the index had just gone through a loop.
- In line 5360, J is the sequential number of the directory entry in the vertical raster. J determines the print position of the entry.
- In line 5380, if the disk number and name were switched for alphabetical sorting, the disk number is returned to the left side of the string.
- Line 6050 right-justifies the program name with spaces or excess characters truncated to a 12-character length. (The maximum length of a TRSDOS file name is 12 characters including the slash and extension of up to three characters. File names read from disks are adjusted to a 12-character length.)
- Line 6080 gives an error message if the program is not in the index.
- Lines 7020–7030 give an error message if there is no index in memory. Number of entries, XX, is saved to disk with the index.
- In line 8020, if V2 = 1, you called a routine to change the index. You are reminded to save the new index before exiting the program.
- Line 10020 pauses until a key is pressed.
- In line 10220, CHR$(2) turns off the cursor to avoid flashes as it moves to the message area.
- In line 10240, Y loops determine on and off time for flashing messages.
- In line 10250, CHR$(1) turns the cursor on.
- In line 10320, line Input avoids printing a question mark on the screen as Input would do.
- Lines 10330–10340 ensure that one-digit disk numbers are preceded by zero, allowing only numbers for disk ID.
- Lines 10410–10490 are a USR routine with supervisory call 53. It reads the entire disk directory into RAM. Each entry is 34 characters. Using PEEK, the name is extracted and right-justified with spaces to a 12-character length. The disk number, DZ$, is add-

See List of Advertisers on Page 563

Listing continued

```
8030 GOSUB 10020:CLS:ON INSTR("MS",C$) GOTO 520,7020
8040 IF C$="E" OR V2=0 THEN CLS:PRINT"*** END OF INDEX PROGRAM *
**":END ELSE M=2:GOSUB 10120:CLS:GOTO 8020
10000 '************************************
10010 'SUBROUTINES
10020 C$=INKEY$:IF C$="" THEN 10020 ELSE RETURN
10100 '************************************
10110 'FLASHING MESSAGES
10120 F$(0)="*** INCORRECT ENTRY--":F$(1)="ONLY 1 TO 8 PLEASE":F
$(2)="ONLY LETTERS IN BRACKETS PLEASE":F$(3)="NO INDEX FILE":F$(
4)="ENTER 2 NUMBERS, NO DECIMALS":F$(5)="INDEX SAVED TO DISK.":F
$(6)="PLEASE TURN ON PRINTER"
10130 F$(7)="THIS INDEX NUMBER ALREADY IN USE.  REENTER.":F$(8)=
"DISK NUMBER OR PROGRAM NOT IN INDEX":F$(9)="DISK DELETED FROM I
NDEX"
10220 PRINT CHR$(2):FOR X=1 TO 2
10230   IF M<5 THEN PRINT@15,F$(0)+F$(M)+" ***"ELSE PRINT@15,"**
* "+F$(M)+" ***"
10240   FOR Y=1 TO 300:NEXT Y:PRINT@15,SPACE$(65):FOR Y=1 TO 300
:NEXT:NEXT X
10250 M=0:PRINT CHR$(1):RETURN
10300 '**********************************
10310 'DISK ID#
10320 CLS:DN$="":PRINT:LINEINPUT"ENTER A 2 DIGIT DISK ID NUMBER:
  ";DN$
10330 IF DN$="M" THEN 520 ELSE IF LEN(DN$)<>2 THEN 10350
10340 FOR X=1 TO 2:IF INSTR("0123456789",MID$(DN$,X,1))=0 THEN 1
0350 ELSE NEXT X:RETURN
10350 M=4:GOSUB 10120:GOTO 10320
10400 '********************************
10410 'READ DIRECTORY FROM DRIVE 1 TO RAM AND EXTRACT FILENAME
10420 V1=0:U2(1)=53:U2(7)=256:U2(3)=-3267:DEFUSR2=VARPTR(U2(0)):
V1=USR2(0)
10430 IF V3=0 THEN XX=1
10440 FOR X=-3267 TO -3 STEP 34:PN$(XX)=""
10450   FOR Y=0 TO 13
10460     IF Y=0 AND OCT$(X)=35 THEN RETURN
10470     IF Y>0 AND OCT$(X+Y)=58 THEN PN$(XX)=MID$(PN$(XX),2,(Y
-1)):L=LEN(PN$(XX))-1:PN$(XX)=PN$(XX)+SPACE$(11-L):DP$(XX)=DZ$+P
N$(XX):GOTO 10490
10480     PN$(XX)=PN$(XX)+CHR$(OCT$(X+Y)):NEXT Y
10490 XX=XX+1:NEXT X:RETURN
10500 '*********************************
10510 'MACHINE LANGUAGE SHELL SORT
10520  J(0)=VARPTR(DP$(1)):J(1)=XX-2:GOTO 10540
10530 J(0)=VARPTR(PN$(1)):J(1)=U-2
10540 DEFUSR5=VARPTR(U5(0)):V9=USR5(VARPTR(J(0))):RETURN
10700 '*********************************
10710 'DELETE AND SORT MARKERS TO END
10720 FOR X=1 TO U-1:MID$(DP$(A(X)),1,2)="ZZ":NEXT
10730 GOSUB 10520:XX=XX-U
10740 XX=XX+1:M=9:CLS:GOSUB 10120:RETURN
10800 '***********************************
10810 'VERTICAL RASTER ROUTINE
10820 W1=(I-1)MOD4:W2=XXMOD4:W3=W1:IF W2<W1 THEN W3=W2
10830 J=W3+W1*INT(XX/4)+INT((I-1)/4)+1:W4=W1+1:RETURN
10900 '******************************
10910 'DISK INPUT OF PRIOR INDEX
10920 OPEN "I",1,"DSKINDEX":PRINT@658,"*** LOADING INDEX ***";:I
NPUT #1,XX
10930 FOR X=1 TO XX:INPUT #1,DP$(X):NEXT:CLOSE:V3=1:CLS:RETURN
11000 '************************************
11010 'ERROR ROUTINES
11020 IF ERR=52 THEN CLS:PRINT"YOU MUST START AGAIN AND OPEN FIL
ES.":PRINT"PRESS <ENTER> TO RETURN TO TRSDOS":PRINT:PRINT S$(1):
PRINT:PRINT S$(3);:LINEINPUT C$:SYSTEM
11030 IF ERR=53 THEN M=3:GOSUB 10120:CLS:RESUME 520
11040 IF ERR=56 THEN CLS:M=6:GOSUB 10120:RESUME 5320
11100 '***********************************
11110 'DATA FOR PEEK
11120 DATA -13023,8797,26623,17441,8830,26625,-15583,8955,26627,
14910,1330,104,-12255,8773,10757,17697,8779,10759,-15583,8959,23
259,26430,-8910,-13990
11200 '***********************************
11210 'DATA FOR MACHINE LANGUAGE SORT
11220 DATA 24013,-6844,-7715,20189,-8958,838,1048,-6695,-15911,3
3,-18688,17133,-13360,-13512,-15079,-7719,-8743,622,26333,-18685
,17133,-9755,-9775,-13560,2183,20189,-8960,326,8645,1,-9755
11230 DATA -6719,-11815,-6887,10705,-8935,94,22237,6401,-10799,6
373,-7924,2273,2293,-13327,10311,6321,6863,17999,9173,9054,-5290
,-6703,9195,9054,-7850,1284,1568,3340,12064,4120,3340,3112,-1687
0
11240 DATA 1568,4899,3333,-6120,7472,-10791,-9787,-7727,-4681,10
322,5054,-9771,-9791,6,782,-7727,-6903,2539,6373,-7752,-10799,17
65,6659,30542,4729,4899,-2288,-13560,2247,-12776
```

ed to each program name. SPACE$ prints the indicated number of spaces. The symbols # (ASCII 35) and : (ASCII 58) are directory and file-name end indicators, respectively.

- Lines 10520–10540 are the machine-language sort. USR5 is called to sort either the entries on one disk or the whole index.
- Lines 10720-10740 change the disk number of all programs to be deleted to ZZ. Index is sorted by disk number, moving these programs to the end. The counter for number of programs is reduced by U, the number of ZZ marked programs. This effectively deletes these programs.
- Lines 10820–10830 contain the vertical raster algorithm. (See the Athanasiou article.) W4 is the column in which a program is printed.
- Line 10930 sets flag V3 to indicate an index is in memory after loading.
- In line 11020, error 52 means no file was open when Basic entered. Error 53 means no previous index was on the disk. Error 56 means the printer output was selected but the printer wasn't on.
- Lines 11220–11240 are a machine-language Shell sort from *Basic Faster and Better* by Lewis Rosenfelder, page 134. It is a very fast sort that swaps pointers rather than strings.

Hints on Using the Program

If you use disk number 1 or 0 solely for your index, all your programs other than the one directory program will be indexed when you have read your other disks with the program. Your directory will be handy for ready reference on your first disk. Otherwise, you will need to use your back-up disk to index your master disk and then copy the augmented DSKINDEX file from the back-up disk to the master.

The slowest part of the program is the Basic conversion of directory entries for alphabetical search. You will locate your file quickest by using function 6 to find your program by name or disk number, or listing all programs by disk number with function 5. Use the alphabetic listing only for a summary printout of all your disks.

The Model II default-length parameter for printout is 60 lines. Three lines are used for heading. You can print 228 programs on an 11-inch sheet. Using 14-inch paper and setting FORMS P = 84 L = 80, you can get 308 programs on a single sheet. ■

Charles Perelman can be reached at 9777 Whilshine Blvd., Suite 700, Beverly Hills, CA 90212

✓See List of Advertisers on Page 563

Compress, Mod II Style

by Charles Wood

Basic programs written with spaces between words, and many comments, are easy to read but quickly eat up memory. Here's the solution.

Recently I entered a long Basic program into my Model II with spaces between words to clearly show the syntax. The listing was generous with comments, even describing how each variable was used.

```
100 'CMPRS -    Compress BASIC statements      by Charles Wood
112 'output file name has suffix '/cmp'
130 CLEAR 1000:DEFSTR B-D:DEFINT E-Z
150 CLS:PRINT"CMPRS - Compress BASIC statements":PRINT:PRINT
160 PRINT"Did you load -F:2?":PRINT
200 INPUT "Name of File ";DI 'Input file name
210 DO=LEFT$(DI,8)+"/cmp" 'Output file name has suffix '/cmp'
220 ON ERROR GOTO 2010 'file not found/wrong mode
300 OPEN "I",1,DI 'read only input
305 ON ERROR GOTO 2030 'no space available
310 OPEN "O",2,DO 'write only output
315 ON ERROR GOTO 2000
320 IC=0:OC=0 'Input/output chrctr counters
400 LINE INPUT#1,BI 'read a statement to BI
405 L=LEN(BI):IC=IC+L 'Length of input statement augments IC
407 QF=0 'quote flag
408 CC=""
410 FOR I=1 TO L
420 G=ASC(MID$(BI,I,1)) 'ASCII of a character
430 IF QF=0 AND G=39 THEN 690 'ends in remark
440 IF QF=0 AND G=34 THEN QF=1:GOTO 480 'start quote
450 IF QF=1 AND G=34 THEN QF=0 'end quote
460 IF QF=1 THEN 480 'enclosed quotes unaltered
470 IF G=32 OR G=9 THEN NEXT I
480 CC=CC+MID$(BI,I,1):NEXT I
490 GOTO 700
690 IF I<8 THEN 400 'delete the line   ;option IF I<8 THEN I=L+1
700 L=LEN(CC):OC=OC+L 'Length of out statement augments OC
750 PRINT CC:PRINT #2, CC 'sequential write one statement
800 GOTO 400 'back for another
2000 IF ERR=60 OR ERR=62 THEN RESUME 2020
2005 PRINT"ERR #";ERR;" at ";ERL:STOP
2010 IF ERR=53 OR ERR=54 OR ERR=63 THEN RESUME 2011 ELSE RESUME
2005
2011 CLOSE:PRINT "File not found or Bad file mode":PRINT:GOTO 20
0
2020 CLOSE:PRINT:PRINT:PRINT"End of File":PRINT:PRINT
2021 PRINT "[";IC;"] input  characters in file ";DI
2022 PRINT "[";OC;"] output characters"
2023 PRINT:PRINT TAB(13) "Saved"; 100*(IC-OC)/IC;"%.":PRINT:GOTO
 200
2030 IF ERR=59 THEN RESUME 2031 ELSE 2010
2031 CLOSE:PRINT"Disk is full":PRINT:GOTO 200
```

Program Listing 1

But the inevitable happened. I finally developed the perfect program, but it couldn't process anything because there was no more work space. The fix was to extract a run-time program that loaded memory without spaces and comments. I needed a compress utility. Although Model III TRSDOS supplies one, Model II TRSDOS does not.

Simply eliminating spaces can be dangerous. For example, compare these two statements:

```
40 IF Q=BF OR A=13 THEN...
50 IFQ=BFORA=13THEN...
```

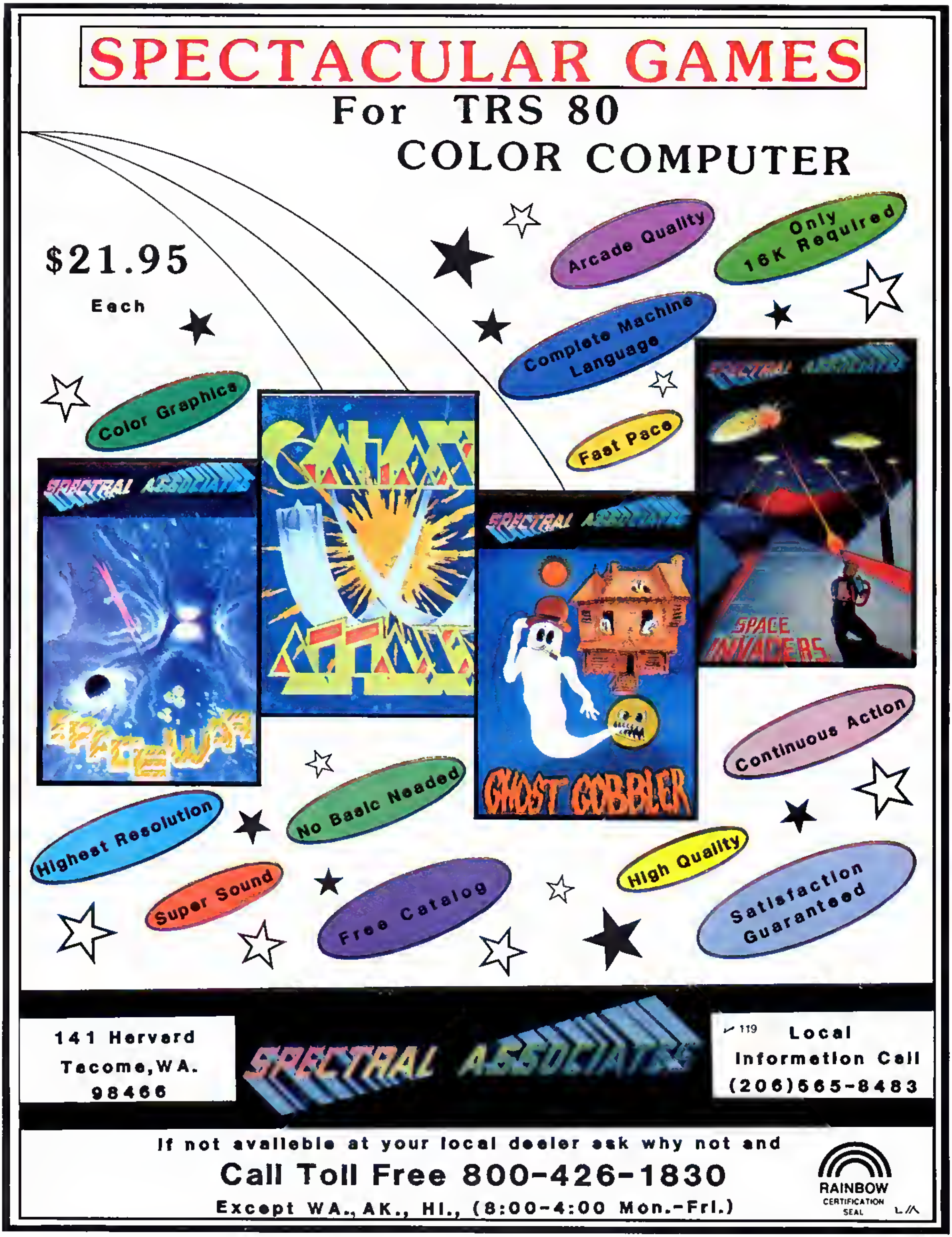
Line 50 is a spaceless version of line 40. Line 50 is syntactically wrong. Basic searches a statement for keywords first, then for variables. Without spaces, the command Or becomes For and the variable BF is broken up. The perfect compress program analyzes syntax as completely as the Basic interpreter itself.

But these combinational errors rarely appear. If one does occur, Basic usually flags an error. Also, programmers are sensitive to the listing of the program and can quickly eyeball a funny-looking statement. Since I check out my own software, I decided to write a short compress program and save money.

My simple approach is to work with a Basic file saved in the ASCII mode (SAVE "filename",A). Here, the record length (LRL) is one and the number of records in the file is the number of bytes in the source program. Files of this kind make disk input/output (I/O) very easy by using the sequential

The Key Box

Model II or 16
32K RAM
Disk Basic
TRSDOS 1.2 or 2.0, 1 disk drive


SPECTACULAR GAMES
For TRS 80
COLOR COMPUTER
$21.95
Each
Arcade Quality
Only 16K Required
Complete Machine Language
Color Graphics
Fast Pace
SPECTRAL ASSOCIATES
SPACE INVADERS
GHOST GOBBLER
Continuous Action
Highest Resolution
No Basic Needed
High Quality
Super Sound
Free Catalog
Satisfaction Guaranteed
141 Harvard
Tacoma, WA.
98466
SPECTRAL ASSOCIATES
119
Local
Information Call
(206)565-8483
If not available at your local dealer ask why not and
Call Toll Free 800-426-1830
Except WA., AK., HI., (8:00-4:00 Mon.-Fri.)
RAINBOW
CERTIFICATION
SEAL

```
130 CLEAR1000:OEFSTRB-D:OEFINTE-Z
150 CLS:PRINT"CMPRS - Compress BASIC statements":PRINT:PRINT
160 PRINT"Did you load -F:2?":PRINT
200 INPUT"Name of File ";OI
210 DO=LEFT$(DI,8)+"/cmp"
220 ONERRORGOTO2010
300 OPEN"I",1,DI
305 ONERRORGOTO2030
310 OPEN"O",2,DO
315 ONERRORGOTO2000
320 IC=0:OC=0
400 LINEINPUT#1,BI
405 L=LEN(BI):IC=IC+L
407 QF=0
408 CC=""
410 FORI=1TOL
420 G=ASC(MID$(BI,I,1))
430 IFQF=0ANDG=39THEN690
440 IFQF=0ANDG=34THENQF=1:GOTO400
450 IFQF=1ANOG=34THENQF=0
460 IFQF=1THEN480
470 IFG=32ORG=9THENNEXTI
480 CC=CC+MID$(BI,I,1):NEXTI
490 GOTO700
690 IFI<8THEN400
700 L=LEN(CC):OC=OC+L
750 PRINTCC:PRINT#2,CC
000 GOTO400
2000 IFERR=60ORERR=62THENRESUME2020
2005 PRINT"ERR #";ERR;" at ";ERL:STOP
2010 IFERR=53ORERR=54ORERR=63THENRESUME2011ELSERESUME2005
2011 CLOSE:PRINT"File not found or Bad file mode":PRINT:GOTO200
2020 CLOSE:PRINT:PRINT:PRINT"End of File":PRINT:PRINT
2021 PRINT"[";IC;"] input  characters in file ";OI
2022 PRINT"[";OC;"] output characters"
2023 PRINT:PRINTTAB(13)"Saved";100*(IC-OC)/IC;"%.":PRINT:GOTO200
2030 IFERR=59THENRESUME2031ELSE2010
2031 CLOSE:PRINT"Oisk is full":PRINT:GOTO200
```

Program Listing 2

read/write statements LINE INPUT #1 and PRINT #2, where #1 and #2 are buffer numbers.

The fancy approach is to work with the non-ASCII saves that contain the numeric code for the command. By logical matching of op-codes and tables, the fancy approach can prevent troubles by leaving some spaces in. But the file structure is harder to work because statements can bridge two records (LRL = 256); deblocking and reblocking becomes awkward. Further, I did not want to invest time for logical tables that have a marginal advantage.

When compressed, the long Basic program of some 13,000 bytes was reduced by 1,600 bytes, a saving of 12 percent. The listing of the long program remains well documented in an auxiliary file. Moreover, the compressed program ran the first time; there were no combination errors.

As shown in the Program Listings, one statement is read at a time. Spaces between quotes are retained (line 460). Other spaces, tabs, and remarks are deleted by making them ASCII nulls (statement 470). The string is then reassembled at which time the nulls are discarded. The statement is then written back to the output file in line 750. Lines 2000 onward trap errors such as end of file and wrong file type.

Some programmers may want to make a change in line 690. If your Basic programs use "GOTO all-remark statements," you may want to preserve the all-remark statements:

```
690 IF I<8 THEN I = L + 1 'retain the statement
```

Program Listing 1 is the CMPRS utility in the fully documented form. Listing 2 is the same program, but compressed. The saving is 32 percent. After compressing a program, load Basic with the compressed version and then save it, this time in the regular non-ASCII format. Check your listing for possible errors and then test.

To run the program from TRSDOS, type BASIC CMPRS –F:2. Then, you can compress programs of your choice. To compress, the input program must be saved in the ASCII mode.

Follow the prompting of CMPRS. The output will be "filename/cmp." You can then load the compressed file by Basic, obtain a listing, and test the program. ■

Charles Wood can be reached at 327 Seventh Ave., Pelham, NY 10803.

RIBBONS

Low Price • FREE Shipping

SATISFACTION GUARANTEED

MX-80 Cartridge

price each in quantity of

1-2	3-5	6-11	12-23	24-47
8.95	8.41	7.90	7.43	6.99

RIBBON LOOPS
new ribbon for your old cartridge
top quality nylon • standard matrix ink

prica aach in quantity of

	1-2	3-5	6-11	12-23	24-47
MX-70/MX-80/MX-100	3.95	3.63	3.34	3.08	2.83
MX-100 double length	7.86	7.23	6.65	6.12	5.63
Decwriter LA34	2.92	2.69	2.47	2.27	2.09
Radio Shack 26-1418	2.26	2.08	1.91	1.76	1.62

New cartridges are available in black ink only.
Loops are available in blue or black ink (same price).
You may mix any combination for quantity price breaks.
Ribbon loops include DO-IT-YOURSELF INSTRUCTIONS.

VISA **DATA SYSTEMS** MasterCard ✓99

(305) 788-2145 • Box 99 • Fern Park, FL 32730

FREE

business software directory

- Radio Shack's Model I, II, III.
- Heath's MBASIC and HDOS
- CPM: Xerox, Alto...
- IBM Personal Computer

"IDM2 is GREAT !" -publisher of 80-US

"(GL) superior to either the Osborne (SBSG & Taranto) or Radio Shack... MAIL-X has a greater capacity... more flexible than (R.S.)"
-columnist of 80-microcomputing

"imperceptively fast...(DBMS) is a good and reliable workhorse" -publisher of Interface Age

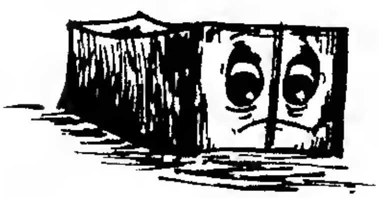
Data base manager, integrated accounting package (AR, AP, GL & Payroll), inventory, word processing, and mailing list. Compare and be selective !

Micro Architect, Inc.
96 Dothan St., Arlington, MA 02174

✓149

TIRED OF NO DISK SOFTWARE?

TRS-80



Flexplus Compatible Software

Sleuth Disassembler - A program which allows user to examine, modify and disassemble object program files on disk or in memory on 6809 systems. Will convert an object file in RAM to a text file on disk

Tabularasa (Electronic spread sheet) - These programs enable the user to generate and maintain tabular computation schemes providing a simple user interface and sophisticated report - generation.

Cross Assembler - A set of modules for 6800/1, 6805, 6802, 8080/5 and Z-80 micro processors which will enable the user to convert a master assembly module to assemble code for any of the above micros

PASCAL

FLEXPLUS DOS

FLEXPLUS is a powerful, easy-to-use disk operating system. Spectral Associates has adapted TSC's FLEX to the best DOS completely compatible with Radio Shack software for use on the Color Computer. Eliminate the need for Radio Shack's TRS DOS use **FLEXPLUS** with Editor Assembler and have the options of a full range of utilities. **FLEXPLUS** works on the 32K Radio Shack disk system with 64K memory chips with a High Resolution multi screen format that supports a 24 line by 51 character display! Also included are special enhancements to Radio Shack's Disk system when you are running FLEX with single or double sided, single or double density, 35, 40 and 80 track drives

Advantages of FLEXPLUS are:

- Best price anywhere
- Easy start-up — just type "RUN FLEX+"
- Allows you to save RS compatible disk files from FLEXPLUS
- All FLEX compatible software will run including INTERRUPT DRIVEN SOFTWARE
- NO HARDWARE MODIFICATIONS NEEDED
- Warranty will not be voided — no need to open computer
- Wide range of available software
- Requires Supercharger board
- $149.95 for **FLEXPLUS**, Editor/Assembler and Supercharger

TOLL FREE
ORDERS ONLY
800-426-1830
EXCEPT WA, HI, AK

SPECTRAL ASSOCIATES

145 Harvard Ave.
Tacoma, Washington 98466
(206) 565-8483

WRITE FOR COMPLETE CATALOG
ADD 3% FOR SHIPPING NO COD
VISA OR MASTERCHARGE ACCEPTED

AVAILABLE ON DISK:

- Editor Assembler
- Super Monitor
- Color Text Star Word Processor
- Sound Source
- Compuvoice

SUPERCHARGER - Allow machine language access to all 64K RAM in a Series E PC board. No hardware mods. $39.95
DISK BASED EDITOR/ASSEMBLER - MACROS, CONDITIONAL ASSEMBLY - will work with Supercharger. $49.95

Call or write us for price information and for a complete list of Flexplus Compatible Software

FANTASTIC BARGAIN

FLEXPLUS, Editor/Assembler and Supercharger
ALL FOR ONLY $149.95

✓119

✓See List of Advertisers on Page 563

Take a Letter. . .

by James Barbarello

Spruce up your printer output with this Basic program for your Model II. It lets you draw large block letters using standard characters.

A common feature of reports generated by data processing facilities is large lettering on title pages. This lettering is usually a 9 by 7 array of standard-size letters forming the large character. The large T is made up of standard size Ts, the large E is made up of standard size Es, and so on.

Letters provides your Model II with this function.

Letters gives you a number of capabilities. You can print any message in large format on any printer (no maximum column restriction). In addition, you can change the letter's shape to your liking. You can also change the standard-size letters that form the characters. Letters can be incorporated into your own programs.

The Strategy

Forming readable characters by arranging printed symbols in an array is far from unique. This is the way characters are formed on a video monitor or a dot-matrix printer. The monitor or printer receives a message indicating the character you want printed and looks for the instructions to form that character in ROM. We'll follow essentially the same strategy, but the ROM will be a disk file, and not an integrated circuit. Using the LETTERS/ROM file makes the program required to do the printing relatively short. It also makes it simple to modify any of the instructions for printing the allowable characters.

The LETTERS/ROM file contains instructions for printing ASCII characters 32 (space) through 90 (Z). This includes all numbers, uppercase letters, and punctuation marks. The first instruction is for a space (ASCII 32), the second is for an exclamation mark (ASCII 33), and so on. Then it's simple to convert the characters to their ASCII equivalent numbers, subtract 31, and access that instruction directly. In its simplest form, the Basic program is about 10 lines. The Letters program is more complex.

The Program

Letters begins by opening the LETTERS/ROM data file and printing the message "Letter Writer" on the screen in large characters (lines 50–80). This is a menu-driven program with four tasks. Tasks 1 and 2 (write message) are similar, task 1 writes to the printer and task 2 writes to the screen. Task 2 is useful if you want to see how a message looks before you print it. Task 3 lets you change the form of the character as well as from which ASCII characters it is formed. Task 4 (end) closes open files automatically. It is good pro-

Program Listing

```
1 REM**          PROGRAM # P4
2 REM**          NAME: LETTERS
3 REM**          LATEST UPDATE: #0, 26 SEPT 1981
4 REM**
5 REM** THIS PROGRAM DRAWS LARGE LETTERS ON SCREEN OR PRINTER
6 REM**
10 CLEAR 3000:CLS:DEFSTR A,D:DIM AM(12,10),A(10),NU(12):LN=0:P=0
:N=0
20 ON ERROR GOTO 5500
30 DLN=STRING$(79,150):DR=CHR$(26):DN=CHR$(25):PRINT@(11,30),DR;
" WAIT... ";DN;
40 OPEN"D",1,"LETTERS/ROM",10:FIELD 1,10 AS AROM
50 MSG$="LETTERWRITER":GOSUB 5000
60 FOR J=1 TO 10:FOR I=1 TO 6
70 PRINT@(J-1,12*(I-1)+5),AM(I,J);:PRINT@(J+10,12*(I-1)+5),AM(I+
6,J)
80 NEXT I,J
90 PRINT@(22,25),DR;" Press ENTER to continue... ";DN;
100 A=INPUT$(1):IF ASC(A)<> 13 THEN 100
110 CLS:PRINT@(4,27),"L E T T E R   W R I T E R":PRINTDLN
120 PRINT@(8,25),DR;" 1 ";DN;" Write to PRINTER"
130 PRINT@(10,25),DR;" 2 ";DN;" Write To SCREEN"
140 PRINT@(12,25),DR;" 3 ";DN;" CHANGE Letters"
150 PRINT@(14,25),DR;" 4 ";DN;" END"
```

Listing continues

The Key Box

Model II or 16
64K RAM
Disk Basic
TRSDOS 2.0A
One Disk Drive
Printer

gramming practice to include an end task in all programs that use disk files.

Task 1 asks for the maximum number of columns on your printer. It then calculates the maximum letters that can be printed on each line (MSGSIZE in line 1040). This information flashes on the screen and a Forms set is performed to reset the computer's line counter. Now you're asked to enter the message. If the message is greater than the allowable maximum, it is erased from the screen and nothing prints. When you enter a printable message, the screen shows a wait prompt while the subroutine at line 5000 constructs the instructions for the printer.

If the printer isn't available, the message "Printer is not available" displays and you are asked if you want to continue or return to the menu. If the printer is available, the message prints on the line printer. Upon completion, you are asked again if you want to continue or return. You can continue printing your message one line at a time until finished, and then return to the menu.

Task 2 is similar to task 1, except that the message is printed on the screen. Because there are only 80 screen columns, the message length is fixed at six characters. If you enter a message greater than six characters, only the first six are displayed on the screen. As with task 1, you can continue printing messages on the screen until you select R (return to menu).

Task 3 allows you to change any character's form in the LETTERS/ROM file. Each character is formed by a 10 by 10 grid of locations. Each location can contain a blank or any printable character. Large characters are formed by arranging printable characters in a recognizable pattern. When you select task 3, the message in lines 3010–3030 flashes on the screen. You're then asked to press the key of the character you want to change the form of.

The LETTERS/ROM file is then accessed and the selected character displays on the screen in a 10 by 10 grid. All blank positions contain a dot (period). Then the message in lines 3120 and 3130 displays to show how to perform modifications. If you press the space bar to blank out a position, a dot appears there.

As indicated in the instructions, the arrow keys move the cursor. Cursor movement is transparent and doesn't disturb the display. When you're finished, press F1 to end the modification procedure. At this point, you have

Listing continued

```
160 PRINT@(22,20),"Press the ";DR;" NUMBER ";DN;" of the Task De
sired...";
170 A=INPUT$(1):IF VAL(A)<1 OR VAL(A)>4 THEN 170ELSE TSK=VAL(A)
180 ON TSK GOSUB 1000,2000,3000,4000:GOTO 110
1000 REM**        PROGRAM BEGINS
1010 CLS:PRINTTAB(30);"TASK # 1: WRITE TO PRINTER":PRINTDLN:GOSU
B 5120
1020 PRINT@(22,0),CHR$(24):PRINT@(3,10),"Enter MAXIMUM Number of
 Columns Available on Printer (EX:80)...";
1030 LINE INPUT A:NU=VAL(A):IF NU<12 OR NU>132 THEN 1020
1040 MSGSIZE=INT(NU/12):TB=(NU-MSGSIZE*12)/2
1050 PRINT@(5,10),"MAXIMUM PRINTER COLUMNS:";DR;NU;DN;TAB(45);"M
AXIMUM LETTERS PER LINE:";DR;MSGSIZE:GOSUB 5120:PRINT@(10,0),;:S
YSTEM"FORMS L=66,W=132":PRINT@(10,0),CHR$(24)
1060 PRINT@(22,0),CHR$(24):PRINT@(10,20),"Enter MESSAGE (";MSGSI
ZE;"characters MAX)...";CHR$(23);
1070 LINE INPUT A:IF LEN(A)>MSGSIZE THEN 1060
1080 PRINT@(22,35),DR;" WAIT... ";DN;:MSG$=A:GOSUB 5000
1090 PRINT@(15,0),CHR$(24):PRINT@(22,35),DR;" WAIT... ";DN;:LPRI
NT CHR$(1):REM* DUMMY LPRINT TO CHECK STATUS OF PRINTER *
1100 FOR J=1 TO 10:FOR I=1 TO LEN(MSG$):LPRINT TAB(TB);AM(I,J);"
 ";:NEXT I:LPRINT:NEXT J:LPRINT:LPRINT
1110 PRINT@(22,0),CHR$(23);:GOSUB 5120:GOTO 1060
2000 CLS:PRINTTAB(30);"TASK # 2: WRITE TO SCREEN":PRINTDLN:GOSUB
 5120
2010 PRINT@(2,0),CHR$(24):PRINT@(4,20),"Enter Message (6 Charact
ers MAXIMUM)...";:LINE INPUT MSG$:IF LEN(MSG$)>6 THEN MSG$=LEFT$
(MSG$,6)
2020 GOSUB 5000:PRINT@(6,0),;
2030 FOR J=1 TO 10:FOR I=1 TO LEN(MSG$):PRINTTAB(5);AM(I,J);"  "
;:NEXT I:PRINT:NEXT J
2040 GOSUB 5120:GOTO 2010
3000 CLS:PRINTTAB(25);"TASK # 3: CHANGE LETTERS":PRINTDLN
3010 PRINT@(3,0),"     This task allows you to change the form o
f the letters to be printed. The
3020 PRINT"change is permanently stored in the LETTERS/ROM Disk
File, which contains the"
3030 PRINT"form of all the printable letters.":GOSUB 5120
3040 GOSUB 5200:PRINT@(3,0),"Press the LETTER you wish to change
...";
3050 A=INPUT$(1):NU=ASC(A):IF NU<32 THEN 3050ELSE IF NU>96 THEN
NU=NU-32:PRINT CHR$(NU) ELSE PRINT A
3060 NU=NU-31:NU=NU*10-9:IF NU>LOF(1) THEN 3060
3070 FOR J=NU TO NU+9:GET 1,J
3080 AM(1,J-NU+1)=AROM:NEXT J
3090 FOR I=10 TO 19:PRINT@(I,35),AM(1,I-9):NEXT
3100 FOR I=1 TO 10:FOR J=1 TO 10:IF MID$(AM(1,I),J,1)=" " THEN P
RINT@(I+9,J+34),".";
3110 NEXT J,I
3120 PRINT@(21,0),"Use the ARROW keys to position the cursor (Fo
reward, Backspace, Up or Down)."
3130 PRINT"Type in new characters as desired (use the Spacebar t
o delete a character).":PRINT"Press ";DR;" F1 ";DN;" when ALL ch
anges have been completed...";:PRINT@(10,35),;
3140 GOSUB 5300:PRINT@(21,0),CHR$(24):PRINT@(21,4),"To save this
 new form, press ";DR;" C ";DN;". Otherwise, press ";DR;" R ";DN
;"...";
3150 GOSUB 5130:PRINT@(20,0),CHR$(24):PRINT@(22,30),DR;" SAVING
NEW FORM... ";DN;
3160 FOR J=NU TO NU+9:LSET AROM=AM(1,J-NU+1):PUT 1,J:NEXT J
3170 PRINT@(22,20),"New Form Saved. Press ANY key to Return to M
enu...";:A=INPUT$(1):RETURN
4000 CLS:PRINT@(0,33),"TASK # 4: END":PRINTDLN
4010 GOSUB 5100
4020 GOSUB 5200:PRINT@(10,35),DR;" WAIT... ";DN;
4030 CLOSE:PRINT@(10,35),DR;" PROGRAM ENDED. ";DN:PRINT@(18,0),;
:CLEAR 50:END
5000 REM**        ACCESS LETTERS/ROM FILE SUBROUTINE
5010 FOR I=1 TO LEN(MSG$):NU(I)=ASC(MID$(MSG$,I,1)):IF NU(I)>96
THEN NU(I)=NU(I)-32
5020 NU(I)=NU(I)-31:NEXT
5030 FOR X=1 TO LEN(MSG$):NU=NU(X)*10-9:IF NU>LOF(1) THEN NU=1
5040 FOR J=NU TO NU+9:GET 1,J
5050 AM(X,J-NU+1)=AROM:NEXT J
5060 NEXT X:RETURN
5070 FOR J=1 TO 10:FOR I=1 TO LEN(MSG$):PRINTAM(I,J);"   ";:NEXT
I:PRINT:NEXT J
5080 IF TTL=1 THEN PRINT:PRINT:PRINT
5100 REM**        CONTINUE/RETURN QUERY (CLEAR TO END OF SCREEN)
5110 PRINT@(2,0),CHR$(24)
5120 PRINT@(22,30),DR;" C ";DN;"ontinue or ";DR;" R ";DN;"eturn.
..";
5130 A=INPUT$(1):NX=ASC(A):IF NX>96 THEN NX=NX-32:A=CHR$(NX)
5140 IF A="R" THEN IF ERR=0 THEN 110ELSE RESUME 110ELSE IF A<>"C
```

Listing continues

Listing continued

```
" THEN 5130
5150 PRINT A;:RETURN
5200 REM**        CLEAR SCREEN BELOW TITLE
5210 PRINT@(2,0),CHR$(24):RETURN
5300 REM**        CHANGE LETTER UTILITY
5310 LN=10:P=35
5320 A=INPUT$(1):N=ASC(A):IF N=1 THEN RETURN ELSE IF N<28 THEN 5
320
5330 IF N<28 THEN 5320ELSE IF N<32 THEN 5370
5340 MID$(AM(1,LN-9),P-34,1)=CHR$(N):PRINT@(LN,P),CHR$(N);:IF N=
32 THEN PRINT@(LN,P),".";
5350 P=P+1:IF P>44 THEN P=35:LN=LN+1:IF LN>19 THEN LN=19
5360 GOTO 5410
5370 IF N=28 THEN P=P-1:IF P<35 THEN P=35
5380 IF N=29 THEN P=P+1:IF P>44 THEN P=35:LN=LN+1:IF LN>19 THEN
LN=19
5390 IF N=30 THEN LN=LN-1:IF LN<10 THEN LN=10
5400 IF N=31 THEN LN=LN+1:IF LN>19 THEN LN=19
5410 PRINT@(LN,P),;:GOTO 5320
5500 REM**        ERROR TRAPPING ROUTINE
5510 IF ERR=56 THEN PRINT@(15,0),CHR$(24):PRINT@(15,28),DR;" PRI
NTER IS NOT AVAILABLE...";DN:GOSUB 5120:RESUME 1090
5520 RESUME 110
```

```
888.......  ...8888...  ...888....  .88888888.  88888888..  .88888888.
.888......  ..888888..  ..88..8...  8888888888  888888888.  8888888888
..888.....  .88....88.  .88...88..  888....888  888....888  888.......
...888....  .88....888  .88.8.888.  888....888  888....888  888.......
....888...  .......88.  88.8.888..  8888888888  888888888.  888.......
....888...  .....888..  88..8.8...  8888888888  888888888.  888.......
...888....  ....88....  .88..8....  888....888  888....888  888.......
..888.....  ....88....  .88.......  888....888  888....888  888.......
.888......  ..........  ..88......  888....888  888888888.  8888888888
888.......  ....88....  ...8888...  888....888  88888888..  .88888888.


8888888...  8888888888  8888888888  ..888888..  888....888  .8888888..
888888888.  8888888888  8888888888  .88888888.  888....888  .8888888..
888....888  888.......  888.......  888.......  888....888  ...888....
888....888  888.......  888.......  888.......  888....888  ...888....
888....888  8888888...  8888888...  888.......  8888888888  ...888....
888....888  8888888...  8888888...  888...8888  8888888888  ...888....
888....888  888.......  888.......  888...8888  888....888  ...888....
888....888  888.......  888.......  888....88.  888....888  ...888....
888888888.  8888888888  888.......  .88888888.  888....888  .8888888..
8888888...  8888888888  888.......  ..88888...  888....888  .8888888..


...8888888  888....888  888.......  888....888  8888....88  ...8888...
...8888888  888...888.  888.......  8888..8888  8888....88  ..888888..
.....888..  888..888..  888.......  88..88..88  88.88...88  .888..888.
.....888..  888.888...  888.......  88..88..88  88.88...88  888....888
.....888..  888888....  888.......  88..88..88  88..88..88  888....888
.....888..  888888....  888.......  88......88  88..88..88  888....888
88...888..  888.888...  888.......  88......88  88...88.88  888....888
88..888...  888..888..  888.......  88......88  88...88.88  .888..888.
.88888....  888...888.  8888888888  88......88  88....8888  ..888888..
..888.....  888....888  8888888888  88......88  88....8888  ...8888...


88888888..  ...8888...  88888888..  ..888888..  888888888.  888....888
888888888.  ..888888..  888888888.  .88888888.  888888888.  888....888
888....888  .88....88.  888....888  888....888  ...888....  888....888
888....888  88......88  888....888  888.......  ...888....  888....888
888888888.  88......88  888888888.  .88888888.  ...888....  888....888
88888888..  88......88  88888888..  .88888888.  ...888....  888....888
888.......  88....8.88  888.888...  .......888  ...888....  888....888
888.......  .88....88.  888..888..  888....888  ...888....  888....888
888.......  ..8888888.  888...888.  .88888888.  ...888....  .88888888.
888.......  ...8888..8  888....888  ..888888..  ...888....  ..888888..


888....888  88......88  888....888  888...888.  8888888888
888....888  88......88  888....888  888...888.  8888888888
888....888  88......88  888....888  888...888.  ......888.
888....888  88......88  .888..888.  .888.888..  .....888..
888....888  88......88  ..888888..  ..88888...  ....888...
888....888  88..88..88  ..888888..  ...888....  ...888....
.888..888.  88..88..88  .888..888.  ...888....  ..888.....
..888888..  88..88..88  888....888  ...888....  .888......
...8888...  8888..8888  888....888  ...888....  8888888888
....88....  888....888  888....888  ...888....  8888888888
```

Figure 1

the option of saving the modified form; C saves the modified form. If you press R, the LETTERS/ROM file stays the same.

Task 4 ends the program by closing the LETTERS/ROM disk file.

The access LETTERS/ROM file subroutine at line 5000 is the workhorse of the program. You can include this subroutine in any of your programs. Lines 5010 and 5020 convert any lowercase characters in the message to uppercase. Lines 5030–5060 put the character information in an array the length of the message by 10 deep. Line 5030 begins incrementing through the message.

Records 1–10 in the LETTERS/ROM file contain the instructions for a space (ASCII 32). Records 11–20 contain the instructions for an exclamation point (ASCII 33), and so on. Line 5030 determines where the information for the nth character in the message begins, and assigns that number to NU. If the character in the message isn't in the LETTERS/ROM file, then NU equals one (and prints a space). Line 5040 gets the instruction for each of the 10 rows, and line 5050 puts that information in the AM array. Once the AM array contains the information for the complete message, program execution returns to the calling line. One program line (in this case, line 1100) prints the complete message.

The remaining subroutines of the program (lines 5070–5520) are program utilities used in the different tasks.

To create the LETTERS/ROM disk file, run the following short program:

```
10 CLEAR 100
20 OPEN"D",1,"LETTERS/ROM",10:
FIELD 1,10 AS A$
30 LSET A$=SPACE$(10)
40 FOR I=1 TO 590:PUT 1,I:NEXT I:
CLOSE
```

This one-time program creates a blank file. Now you can use task 3 in the Letters program to form the 58 large characters (ASCII 33 through ASCII 90). ASCII 32 (space) was formed when the blank file was created. You can form each large character exactly as shown in Fig. 1, or modify the Fig. 1 ROM set to your own desires. As you build your ROM file, you can periodically check its contents using task 2 (write to screen). ■

Jim Barbarello lives at RD #1 Box 241H, Tennent Road, Englishtown, NJ 07726. He enjoys woodworking, music, and computing.

PANIK

(BY WILLIAM DEMAS)

- MACHINE LANGUAGE
- ARCADE STYLE GAME
- JOY STICK COMPATIBLE
- SOUND
- SUPER ENHANCED DISK VERSION
- 10 HIGH SCORE SAVE
- HIGH RESOLUTION GRAPHICS

All Sound effects including voice are output through cassette aux plug.

The arcade game of the century. It has a little of all arcade games built in but it is its own original game. If you like arcade games with a lot of action and excitement, this one is a must.

(SEE YOUR LOCAL DEALER) ~36

● ● OR ORDER FROM ● ●

FANTASTIC SOFTWARE

P.O. Box 27734

Las Vegas, Nevada 89126

(702) 362-1457

MASTER CHARGE VISA

PRICE** 16K TAPE ($19.95)**

32K DISK ($24.95)

MODEL I & III

TAPE VERSION WILL NOT TRANSFER TO DISK

~See List of Advertisers on Page 563

TRUNKS FOR THE MEMORIES.

Introducing the most logical place to store Elephant Memory Systems® (or lesser brands of disks): The Trunk.

With its alphabetized library index, you can file or retrieve up to 60 disks, instantly.

The Trunk is made of durable molded plastic with a hinged, one-piece lid, to keep disks safe from dust, dirt, and other detriments which disks despise.

And, it's portable. Because the lid doubles as a carrying handle so your Elephant Memory Systems® disks can go anywhere you do.

There's a model for 5¼" and 8" floppies, as well as a cassette-and-game file and a special Atari® version.

So if you're looking for the best disk storage system on the market . . .

The Trunk is an open-and-shut case.

THE TRUNK. ENDORSED BY ELEPHANTS.

Elephant Memory Systems® Disks

A full line of top-quality floppies, in virtually every 5¼" and 8" model, for compatibility with virtually every computer on the market. Guaranteed to meet or exceed every industry standard, certified 100% error-free and problem-free, and to maintain its quality for at least 12 million passes (or over a life-time of heavy-duty use).

Marketed exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021. Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.

✓ 472

READER SERVICE

Please help us to bring you a better magazine—by answering these questions.

This card valid until January 31, 1983

A. What type of TRS-80 do you own?
1. Model I Level I
2. Model I Level II
3. Model II
4. Model III Level I
5. Model III Level II
6. Color
7. Pocket

B. On a scale of 1 (little interest) to 5 (great interest), please rate your interest in the following types of articles
___ 1. Product reviews
___ 2. Programming tutorials
___ 3. Construction projects
___ 4. Interviews with industry leaders
___ 5. Industry news
___ 6. Graphics tutorials

C. Do you ever do any consulting towards business purchases of computers?
1. Yes
2. No

D. If you have a computer besides your TRS-80, what kind is it?
1. PMC/LNW
2. Apple
3. IBM
4. Atari
5. Sinclair/Timex
6. Commodore/PET
7. Heath
8. Other ________

E. Where did you buy your TRS-80?
1. Radio Shack store
2. Other computer store
3. Mail order
4. Other ________

F. How much did you save from list price when purchasing your system?
1. Nothing
2. Under $300
3. $300-$600
4. $600-$900
5. Over $900

G. Which of the following do you rate as most important in choosing a vendor?
1. Technical expertise
2. Service capability
3. Prices
4. Location
5. Other ________

H. Which of the following magazines do you read regularly?
1. Microcomputing
2. Desktop Computing
3. Creative Computing
4. InfoWorld
5. Personal Computing
6. 80 US
7. Byte
8. Popular Computing
9. Other ________

I. What is your annual household income?
1. Under $20,000
2. $20,000-$40,000
3. $40,000-$60,000
4. $60,000-$80,000
5. $80,000-$100,000
6. Over $100,000

J. What ads in 80 Micro do you find most interesting?
1. Almost all ads
2. About half the ads
3. Mostly full page ads
4. Ads for products of specific interest
5. Don't look at ads

K. What magazines are more helpful to you than 80 Micro?
____

L. On a scale of 1 (no interest) to 5 (great interest), please rate your interest in the following regular columns:
___ 1. 80 Remarks
___ 2. Money DOS
___ 3. Soft Bits
___ 4. News
___ 5. Feedback Loop
___ 6. Beneath the Keyboard
___ 7. 80 Applications
___ 8. Copernica
___ 9. Medical Opinion
___ 10. EXclusive ORacle
___ 11. Fun House
___ 12. Kitchen Table Software

Reader Service: To receive more information from any of the advertisers in this issue of *80 Microcomputing*, circle the number on the postage-paid Reader Service Card that corresponds with the Reader Service number on the ad in which you are interested. You will find numbers, preceded by a ✓, near the logo of each advertiser. Complete the entire card, drop into a mailbox and in 4-6 weeks you will hear from the advertiser directly.

1	6	11	16	21	151	156	161	166	171	301	306	311	316	321	451	456	461	466	471
2	7	12	17	22	152	157	162	167	172	302	307	312	317	322	452	457	462	467	472
3	8	13	18	23	153	158	163	168	173	303	308	313	318	323	453	458	463	468	473
4	9	14	19	24	154	159	164	169	174	304	309	314	319	324	454	459	464	469	474
5	10	15	20	25	155	160	165	170	175	305	310	315	320	325	455	460	465	470	475
26	31	36	41	46	176	181	186	191	196	326	331	336	341	346	476	481	486	491	496
27	32	37	42	47	177	182	187	192	197	327	332	337	342	347	477	482	487	492	497
28	33	38	43	48	178	183	188	193	198	328	333	338	343	348	478	483	488	493	498
29	34	39	44	49	179	184	189	194	199	329	334	339	344	349	479	484	489	494	499
30	35	40	45	50	180	185	190	195	200	330	335	340	345	350	480	485	490	495	500
51	56	61	66	71	201	206	211	216	221	351	356	361	366	371	501	506	511	516	521
52	57	62	67	72	202	207	212	217	222	352	357	362	367	372	502	507	512	517	522
53	58	63	68	73	203	208	213	218	223	353	358	363	368	373	503	508	513	518	523
54	59	64	69	74	204	209	214	219	224	354	359	364	369	374	504	509	514	519	524
55	60	65	70	75	205	210	215	220	225	355	360	365	370	375	505	510	515	520	525
76	81	86	91	96	226	231	236	241	246	376	381	386	391	396	526	531	536	541	546
77	82	87	92	97	227	232	237	242	247	377	382	387	392	397	527	532	537	542	547
78	83	88	93	98	228	233	238	243	248	378	383	388	393	398	528	533	538	543	548
79	84	89	94	99	229	234	239	244	249	379	384	389	394	399	529	534	539	544	549
80	85	90	95	100	230	235	240	245	250	380	385	390	395	400	530	535	540	545	550
101	106	111	116	121	251	256	261	266	271	401	406	411	416	421	551	556	561	566	571
102	107	112	117	122	252	257	262	267	272	402	407	412	417	422	552	557	562	567	572
103	108	113	118	123	253	258	263	268	273	403	408	413	418	423	553	558	563	568	573
104	109	114	119	124	254	259	264	269	274	404	409	414	419	424	554	559	564	569	574
105	110	115	120	125	255	260	265	270	275	405	410	415	420	425	555	560	565	570	575
126	131	136	141	146	276	281	286	291	296	426	431	436	441	446	576	581	586	591	596
127	132	137	142	147	277	282	287	292	297	427	432	437	442	447	577	582	587	592	597
128	133	138	143	148	278	283	288	293	298	428	433	438	443	448	578	583	588	593	598
129	134	139	144	149	279	284	289	294	299	429	434	439	444	449	579	584	589	594	599
130	135	140	145	150	280	285	290	295	300	430	435	440	445	450	580	585	590	595	600

Name ____________________

Address ____________________

City __________ State _____ Zip _____

80 Micro • Anniversary 1983

BOOKS

80 Microcomputing
Peterborough NH 03458

Please send me the following 80 Microcomputing products:

Qty.	Catalog #	Title	Unit Price	Total

Shipping and handling charges:
$1.50 1st book, $1.00 each additional book
(UPS, use street address)
$10.00 each book overseas airmail

Shipping/handling ________

Total ________

Enclosed $ ________ ☐ Check ☐ M.O.
Bill: ☐ AE ☐ MC ☐ VISA
Card # ____________________
Exp date ________ Interbank # ________
Signature ____________________
Name ____________________
Address ____________________
City __________ State _____ Zip _____

Please allow 4–6 weeks for delivery

No C.O.D. orders accepted.

AN-83

SUBSCRIPTION

80 micro
subscribers save $10 off the newsstand price.

☐ New subscription ☐ Renewal
☐ 1 year—$24.97

Enclosed $ ________ ☐ Check ☐ M.O.
Bill: ☐ MC ☐ Visa ☐ AE ☐ me

Card # ____________________ Exp. date ________
Signature __________ Interbank # ________
Name ____________________
Address ____________________
City __________ State _____ Zip _____

Canada—$27.97, 1 year only, US funds.
Foreign Surface—$44.97 1 year only, US funds.
Foreign Air—Please inquire.

331R8A

Place
Stamp
Here

80micro

P. O. BOX 306
DALTON, MA 01226

Place
Stamp
Here

80micro
Subscription Dept.
POB 981
Farmingdale NY 11737

Place
Stamp
Here

80micro
Att. Mail Order
Peterborough NH 03458

80 LIST OF ADVERTISERS

For further information from our advertisers, please use the Reader Service cerd.

Investment Advisor

by Charles Perelman

When you need to analyze or arrange a financial transaction, IRR (Internal Rate of Return) will help you compare alternative opportunities and financing terms. IRR is the net interest rate you will receive on your invested dollars during the investment period.

Banks and other financial institutions have used IRR to determine yields, generally without considering tax effects. IRR computes the yield to maturity on bonds or mortgages purchased at either a premium or discount rate. Allowances for income taxes must be made before determining the net cash flow for IRR.

Determine the profit margin of an investment by computing its internal rate of return.

Although IRR is primarily applicable to real-estate transactions, it can be used to evaluate any type of investment. Most tax shelters either require that you make a certain number of payments into them, or they pay out several years after the initial investment (often in installments), or both. Syndicated real-estate, oil, and gas and leasing partnerships emphasize tax benefits, and they generally furnish all the data required for IRR computations.

Using the Program

Two types of IRR computations are provided. After entering the cash flows, you can choose your required or minimum IRR, and the program will calculate the present value of the investment at this interest rate. If the IRR value is below the asking price, you can turn down the deal or renegotiate the price to obtain the desired yield. Alternatively, for a given price and terms, you can determine the IRR for cash flows to be received.

If you are contemplating the purchase of two assets with widely varying terms, the program enables you to compare them on the basis of the effective compounded interest earned on your funds. You need to figure the cash requirements and positive cash flow for each year you will hold the investment.

Figure 1 is a worksheet for a shopping center syndication and Fig. 2 is for a corporate jet-aircraft lease deal. The right column gives you the figures for the IRR program. These are actual tax

SHOPPING CENTER

Year	Purchase Installments	Cash Flow	Net Cash Required	Tax Gain Or Loss	Tax Benefit	Net Cash Flow
					50% rate	
0	11700		11700			(11700)
1	17200	175	17025	(29991)	14996	(2029)
2	16575	250	16325	(43021)	21511	5186
3	15600	250	15350	(41453)	20727	5377
4	14975	250	14725	(39059)	19530	4805
5	13700	375	13325	(37491)	18746	5421
6		375	(375)	(34290)	17145	17520
Sale		89750	(89750)	225305	(45061)	44689
				20% capital gain rate		

Internal Rate of Return = 44.38%

Figure 1

CORPORATE JET

Year	Purchase Installments	Cash Flow	Net Cash Required	Tax Gain Or Loss	Tax Benefit	Net Cash Flow
					55% rate	
0	9150		9150			(9150)
1	30230	59	30171	(27476)	15112	(15059)
2	23500	55	23445	(64833)	35658	12213
3	20290	55	20235	(44026)	24214	3979
4	19120	55	19065	(38437)	21140	2075
5		55	(55)	(35578)	19568	19623
6		55	(55)	(13346)	7340	7395
Sale		80150	(80150)	80150	(44083)	36067
				55% ordinary income rate		

Internal Rate of Return = 35.35%

Figure 2

The Key Box

Model II or 16
64K
Disk Basic
TRSDOS 1.2 or 2.0, disk drive

shelters merchandised in 1981.

Tax losses result from a combination of interest paid and depreciation. Cash flows from tax benefits are computed by multiplying gain or loss by your assumed top tax bracket. Fifty percent was used for the real-estate deal and 55 percent for the airplane (based on combined federal and state rate). In the year of sale, tax must be paid on the gain. The maximum long-term capital gains rate of 20 percent applies to the real estate, but depreciation on the jet will be "recaptured," so that gain is taxable at ordinary rates.

In both cases, cash paid out to you during the life of the investment is nominal. Sales price is assumed to be sufficient to return the principal cash invested. Even so, a high rate of return results from "tax leverage." This is an after-tax rate.

The program lets you change the initial cost, sales price, or flow for any year without re-entering the other data. This permits you to explore creative financing terms and assumptions about the state of the market when you dispose of the asset. You can study their effects on the yield of a down payment split over several years, greater purchase price in exchange for a longer due date on junior financing, lower down payment or lower annual payments with greater balloon, and any other variations.

In Fig. 3, calculations are made for three possible sales prices. Amounts

APARTMENT BUILDING

Year	Purchase Installments	Cash Flow	Net Cash Required	Tax Gain Or Loss	Tax Benefit	Net Cash Flow
					55% rate	
0	350000					(350000)
1		8223	(8223)	(84322)	46377	54600
2		9010	(9010)	(67180)	36949	45959
3		14505	(14505)	(53295)	29312	43817
4		20380	(20380)	(38990)	21445	41825
5		26665	(26665)	(24230)	13327	39992
Cash to loan sale: 25% capital gains tax						
A. 7%/yr. appreciation			(689691)	686491	(171623)	518068
B. 3%/yr. appreciation			(470166)	466966	(116742)	353424
C. -0- appreciation			(311200)	368000	(92000)	219200

Internal Rate of Return:
A. IRR = 19.77%
B. IRR = 13.32%
C. IRR = 6.48%

Figure 3

Program Listing

```
100 REM ************************
110 REM INVESTMENT ADVISOR
120 REM BY CHARLES R. PERELMAN, 1982
130 REM 9777 WILSHIRE, BEVERLY HILLS, CA.
140 REM PRESENT VALUE COMPARISON OF INVESTMENTS
150 REM INTERNAL RATE OF RETURN
200 REM ************************
210 DIM F(20),PV(20)
220 D1$="$$#,###,###.##":R$="####.##%":E$="-.1234567890"
250 CLS:PRINTTAB(20)"INVESTMENT ADVISOR":PRINT:PRINT"DO YOU WANT
 INSTRUCTIONS?  (Y OR N)";
260 Y$=INKEY$
270 Z$=INKEY$:IF Z$="" THEN 270
280 IF Z$<>"Y" AND Z$<>"y"AND Z$<>"N" AND Z$<>"n" THEN 270
290 IF Z$="Y" OR Z$="y" THEN GOSUB 6020
300 REM ************************
310 REM MAIN MENU
320 CLS:PRINTTAB(20)"INVESTMENT ADVISOR":PRINT
330 PRINT:PRINT:PRINT"MAIN MENU:":PRINT"   1. ENTER ITEMS":PRINT"
  2. REVIEW OR CHANGE ENTRIES":PRINT"   3. COMPUTE PRESENT VALUE"
:PRINT"  4. COMPUTE INTERNAL RATE OF RETURN":PRINT"  5.  EXIT FR
OM PROGRAM":PRINT:PRINT"     PRESS NUMBER FOR CHOICE";
```

Listing continues

Listing continued

```
340 Y$=INKEY$
35Ø Z$=INKEY$:IF Z$="" THEN 35Ø
36Ø IF VAL(Z$)<1 OR VAL(Z$)>5 THEN L=91Ø:A=1:GOSUB 712Ø:GOTO 34Ø
37Ø ON VAL(Z$) GOSUB 1Ø2Ø,2Ø2Ø,3Ø2Ø,4Ø2Ø,5Ø2Ø
38Ø GOTO 32Ø
1ØØØ REM ********************
1Ø1Ø REM ITEM ENTRY ROUTINE
1Ø2Ø CLS:T=Ø:C=Ø:SP=Ø:FOR X=1 TO N:F(X)=Ø:NEXT:PRINT"ENTER COST
OF ASSET  ";:L=21:PF=1:GOSUB 752Ø:C=VAL(Z1$):PF=Ø
1Ø3Ø IF C=Ø THEN A=5:GOSUB 712Ø:GOTO 1Ø2Ø
1Ø4Ø LINEINPUT"ENTER NUMBER OF YEARS, MAXIMUM OF 2Ø:  ";N$:N=VAL
(N$):L=Ø
1Ø5Ø FOR Y=1 TO LEN(N$):Z=ASC(MID$(N$,Y,1)):IF Z<48 OR Z>57 THEN
 PRINT@119,SPACE$(41);:A=2:L=8Ø:GOSUB 712Ø:N=Ø:N$="":GOTO 1Ø4Ø E
LSE NEXT Y
1Ø6Ø IF N<1 OR N>2Ø THEN PRINT@119,SPACE$(41);:A=2:L=8Ø:GOSUB 71
2Ø:N=Ø:GOTO1Ø4Ø
1Ø7Ø PRINT:PRINT:FOR X=1 TO N:PRINT X;" ENTER ANNUAL CASH FLOW
";:L=(X*8Ø)+268:GOSUB 752Ø:F(X)=VAL(Z1$):NEXT
1Ø8Ø PRINT"ENTER NET SALES PROCEEDS AT END OF LAST PERIOD  ";:PF
=1:L=(X*8Ø)+288:GOSUB 752Ø:SP=VAL(Z1$):PF=Ø
1Ø9Ø RETURN
2ØØØ REM *******************
2Ø1Ø REM REVIEW AND CHANGE
2Ø2Ø IF C=Ø THEN A=5:GOSUB 712Ø:RETURN
2Ø3Ø CLS:PRINT"COST OF ASSET ";:PRINTUSING D1$;C
2Ø4Ø FOR X=1 TO N:PRINT X;" CASH FLOW FOR PERIOD   ";:PRINTUSING
 D1$;F(X):NEXT
2Ø5Ø PRINT"SALES PROCEEDS AT END OF LAST PERIOD  ";:PRINTUSING D
1$;SP
2Ø6Ø PRINT@176Ø,"ENTER NUMBER TO CHANGE, C FOR COST, S FOR SALES
 PRICE, Ø FOR NONE.    ";CHR$(8)+CHR$(8);:LINEINPUT Z$
2Ø7Ø IF Z$="Ø" THEN RETURN
2Ø8Ø IF Z$="C" OR Z$="c" THEN PF=1:L=15:PRINT@L,SPACE$(15);STRIN
G$(15,8);:GOSUB 752Ø:C=VAL(Z1$):PF=Ø:GOTO 2Ø6Ø
2Ø9Ø IF Z$="S" OR Z$="s" THEN PF=1:L=((N+1)*8Ø+39):PRINT@L,SPACE
$(15);STRING$(15,8);:GOSUB 752Ø:SP=VAL(Z1$):PF=Ø:GOTO 2Ø6Ø
```

Listing continues

were adapted from an actual apartment project. Commissions and costs of sale are estimated at six percent of the sale price. Total cost is $1,000,000 with $350,000 down. Depreciation is computed under the new ACRS rules. Comparison for a range of sales prices can be made rapidly with the facilities of the program.

In evaluating real estate, you can compare the results of possible changes in cash flow upon your ultimate return within the range you would anticipate on a worst-case or best-case basis. This is called sensitivity analysis. This kind of calculation is often revealing and can either warn you that the deal is on "thin ice" or else spot a bonanza where you have figured out a way to increase income levels and cash flows.

Perform the following steps:

- Complete your cash-flow worksheet for a property using the current income level with future adjustments for inflation and other pertinent factors.
- Determine the range within which you think income might vary, adjusting annual cash flow accordingly.
- Make a separate worksheet for the top and bottom of the range.
- Run the IRR program for all three

computations. You might be surprised at the difference two or three percent in gross income will make in your effective return.

IRR is also used to determine the best time to sell a property. The interplay among annual tax benefits (which decrease as depreciation is used up and the principal portion of loan payments grows larger), possible yearly increases in value of the asset, and the tax you will have to pay to realize the sales price (which depends upon whether the gain qualifies for long term capital treatment) is taken into account when you determine the net cash flow after taxes and measure the success of your investment with IRR.

Program Structure

Investment Advisor can be converted for use on Model I or Model III with changes in the Print statements. SPACE$ on the Model II prints a series of blanks and can be replaced by STRING$. By dimensioning the program for 20 periods, all inputs fit on the Model II screen and permit review of all data when making changes for subsequent runs.

You can use the same data to deter-

Listing continued

```
2100 IF VAL(Z$)>=1 AND VAL(Z$)<=N THEN X=VAL(Z$):L=((X*80)+27):P
RINT@L,SPACE$(15);STRING$(15,8);:GOSUB 7520:F(X)=VAL(Z1$):GOTO 2
060
2110 A=7:GOSUB 7120:GOTO 2060
3000 REM *****************
3010 REM PRESENT VALUE PROGRAM
3020 CLS:PRINT"ENTER REQUIRED INTEREST RATE WITHOUT PERCENT SIGN
, E.G.":LINEINPUT"FOR SIX AND ONE HALF PERCENT ENTER 6.5  ";Z$
3030 IF VAL(Z$)<=0 THEN A=4:GOSUB 7120:GOTO 3020 ELSE I1=VAL(Z$)
/100
3040 F(N)=F(N)+SP:FOR X=1 TO N:PV(X)=F(X)/((1+I1)^X):T=T+PV(X):N
EXT:F(N)=F(N)-SP
3050 IF T=0 THEN A=5:GOSUB 7120:RETURN
3060 CLS:PRINT"AT INTEREST RATE OF ";USING R$;I1*100:PRINT"PRESE
NT VALUE OF CASH FLOWS IS";:PRINTUSING D1$;T:PRINT
3070 T=0:LINEINPUT"PRESS ENTER TO CONTINE";Z$:RETURN
4000 REM ******************
4010 REM INTERNAL RATE OF RETURN
4020 CLS:PRINT"COMPUTING INTERNAL RATE OF RETURN":I1=.4:I2=0:I3=
2:F(N)=F(N)+SP
4025 IF I1<.01 THEN PRINT:PRINT"SORRY, IRR IS LESS THAN 1%.  BAD
 INVESTMENT OR ERROR?":PRINT"PLEASE START AGAIN":GOTO 4090
4030 FOR X=1 TO N:PV(X)=F(X)/((1+I1)^X):T=T+PV(X):NEXT
4040 IF T=0 THEN A=5:GOSUB 7120:RETURN
4050 IF ABS(C-T)<C*.001 THEN 4080
4060 IF C>T THEN I3=I1 ELSE I2=I1
4070 I1=(I3+I2)/2:T=0:GOTO 4025
4080 PRINT:PRINT"INTERNAL RATE OF RETURN = ";:PRINTUSING R$;I1*1
00
4090 F(N)=F(N)-SP:T=0:PRINT:LINEINPUT"PRESS ENTER TO CONTINUE";Z
$:RETURN
5000 REM *******************
5010 REM END PROGRAM
5020 A=6:PRINT CHR$(31);CHR$(26);:GOSUB 7120:PRINT CHR$(25);CHR$
(30):END
6000 REM *******************
6010 REM INSTRUCTIONS
```

Listing continues

✓See List of Advertisers on Page 563

Listing continued

```
6020 CLS:PRINTTAB(20)"INVESTMENT ADVISOR":PRINT
6030 PRINT"   THE PRESENT VALUE PROGRAM WILL COMPUTE THE VALUE O
F ALL OF CASH FLOWS YOU":PRINT"WILL RECEIVE FROM AN INVESTMENT A
T THE INTEREST RATE OF YOUR CHOICE.  THIS IS":PRINT"THE VALUE TO
 YOU NOW OF THIS INVESTMENT."
6040 PRINT"   THE INTERNAL RATE OF RETURN PROGRAM COMPARES ALL O
F THE CASH FLOWS TO THE":PRINT"AMOUNT OF YOUR INITIAL INVESTMENT
 AND GIVES YOU THE COMPOUND RATE YOUR":PRINT"INVESTMENT EARNS OV
ER ALL THE PERIODS."
6050 PRINT"   ENTER NUMBERS WITHOUT DOLLAR SIGN OR COMMAS.  ENTE
R NEGATIVE CASH FLOWS":PRINT"OR PAYMENTS REQUIRED WITH A MINUS S
IGN.":PRINT:LINEINPUT"PRESS ENTER TO CONTINUE";Z$:RETURN
7000 REM ***********************
7010 REM SUBROUTINES
7100 *************************
7110 REM FLASHING MESSAGES
7120 M$(1)="FROM 1 TO 5 ONLY PLEASE":M$(2)="MUST BE BETWEEN 1 AN
D 20":M$(3)="REENTER---EXCEEDS LIMIT":M$(4)="MUST BE POSITIVE NU
MBER":M$(5)="DATA NEEDED":M$(6)="BYE NOW!!":M$(7)="INCORRECT ENT
RY"
7130 FOR W=1 TO 2:PRINT@45,"*** "+M$(A)+" ***":FOR Y=1 TO 500:NE
XT:PRINT@45,SPACE$(35):FOR Y=1 TO 300:NEXT:NEXT W:PRINT@L,"";:RE
TURN
7200 REM *********************
7500 REM ***********************
7510 REM ENTRY ROUTINE
7520 Y$=INKEY$:Z1$=""
7530 Z$=INKEY$:IF Z$="" THEN 7530
7540 IF ASC(Z$)=8 AND LEN(Z1$)>0 THEN PRINT CHR$(8);:Z1$=LEFT$(Z
1$,LEN(Z1$)-1):GOTO 7530
7550 IF ABS(VAL(Z1$))>9999999.99 THEN A=3:GOSUB 7120:PRINT@L,SPA
CE$(10);STRING$(10,8);:Z1$="":GOTO 7520
7560 IF PF=1 AND Z$="-" THEN A=4:GOSUB 7120:Z1$="":GOTO 7520
7570 IF ASC(Z$)=13 THEN PRINT STRING$(LEN(Z1$),8);:PRINTUSINGD1$
;VAL(Z1$):RETURN
7580 IF INSTR(E$,Z$)=0 OR LEN(Z1$)>0 AND Z$="-" THEN Z$="":GOTO
7530
7590 Z1$=Z1$+Z$:PRINT Z$;:GOTO 7530
```

mine present value of cash flows or IRR. An assumed cost for the investment must be entered. Extensive error trapping is provided. Only positive inputs are accepted for cost and sales price.

The IRR calculation beginning at line 4010 uses a trial-and-error repetitive loop to zero in on the actual rate of return. CHR$(31) changes to 40 characters per line and CHR$(26) to reverse video. CHR$(25) and CHR$(30) return to normal video and size after the program-ending message.

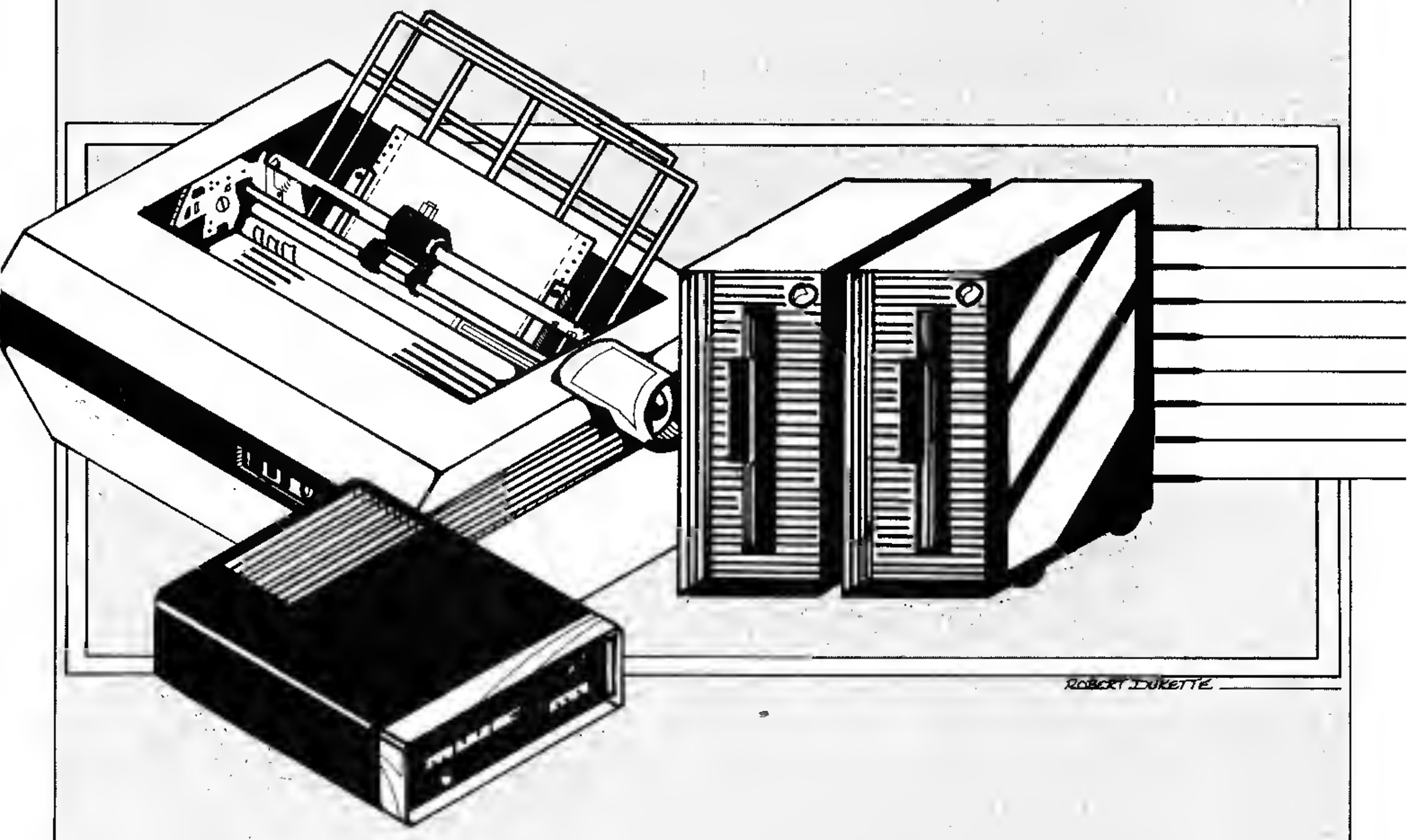
Variable L is used for cursor positioning after error messages. The INKEY$ entry module at 7510 can be used in any program requiring numerical entries. When PF = 1, a minus sign will not be accepted.

Financial return is only one of the measures for sophisticated investing, but it must be evaluated. Use it to help you make intelligent decisions for a successful economic future. ■

Charles R. Perelman is an attorney, CPA and certified tax specialist in California. He lives at 9777 Wilshire Blvd., Suite 700, Beverly Hills, CA 90212.

Part V.

PERIPHERALS



How to Buy a Printer

by Jim Hansen

Epson, IDS, Okidata, NEC—which do you choose? Jim Hansen is an industry insider who can give you good advice when printer shopping.

You've paid your dues. With upwards of $2,000 invested, your computer system has a reasonable amount of memory, disks, and a good monitor. The time has come to select a printer. The choice can be difficult because printer manufacturers do not all talk the same language (from both a technical and ethical point of view), and prices range from around $300 to over $10,000. By selecting the wrong printer you can make an expensive mistake.

The two major printer technologies available are impact (where something strikes the paper to form the character image) and nonimpact printing where heat, light, electric current flow, or other techniques are used to make an image visible.

Generally, nonimpact printing requires the use of special paper. This paper may be coated with aluminum (which is burned away to form characters) or a chemical that is activated by heat or light. Another nonimpact printing technology involves squirting ink directly onto paper. The ink is deflected electrostatically much like the electron beam of a tv picture tube to write the character, or squirted from nozzles to make dot-matrix-like characters. This technology is currently out of the price range of most home computerists.

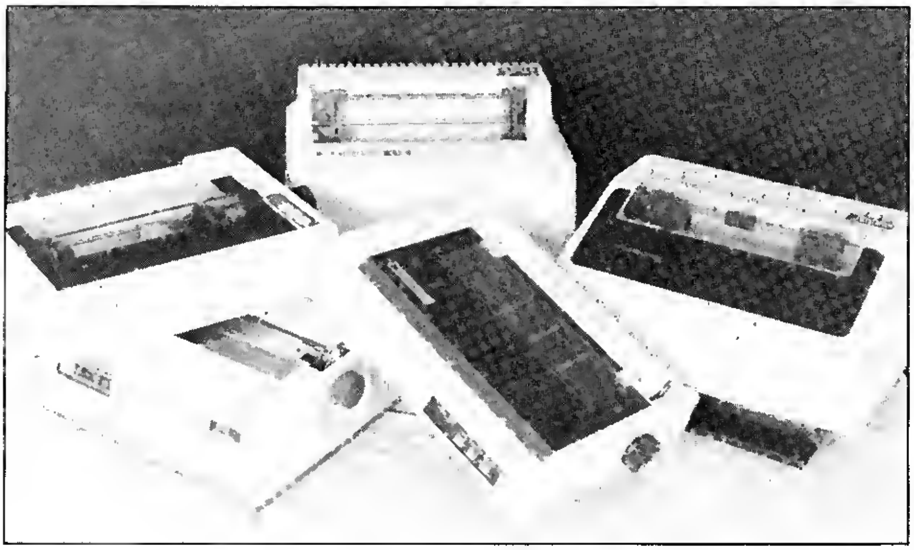


Photo 1. Several popular printers—which one is for you? Top to bottom, left to right, they are the Epson MX-80; the Okidata Microline 82; the IDS 440, the latest low-cost printer from IDS, the Microprism; and finally the full-size Prism 80.

Nonimpact printing is usually in dot-matrix form, characters formed by a series of dots much like a baseball scoreboard. Paper for these printers is available from a limited number of outlets or mail order, and is relatively expensive. I don't like the feel of any of these papers. They tend to curl and never lie flat. Since they come only in rolls, you have to carefully trim the stuff if you want an 8½-by-11 sheet. They are typically thin, and therefore fragile. The images formed by this technology often have low contrast (they aren't very dark) but they usually copy well. Thermal papers (such as used by Texas Instruments and Computer Devices Inc. portable terminals) fade if left in the sun.

The real advantage of nonimpact printing is that it is quiet. I once built an interface for a Casio ink-jet printer. The most noise I heard from the mechanism was the sound of the paper sliding down the back of it during line feeds.

Impact Printers

Impact printing includes all the common dot-matrix, daisy-wheel, and line printers found in computer centers today. The two divisions within this technology are dot-matrix and fully formed character printers. Fully formed character printers use a type pallet much like a typewriter to make the characters. The daisy-wheel and selectric-like (ball) printers fall into this category.

Line printers found in large computer installations have drums, chains, or bands that carry the typeface pallets. Some popular terminals also use this

technique. The General Electric Terminet series is one example. Line printers are usually designed for speed rather than quality. The daisy-wheel and ball printers are designed to produce quality print.

Selectric-based printers use a variation of the IBM Selectric typewriter mechanism (usually called a Selectric I/O model) and can print around 10 to 15 characters per second. Output from a well adjusted Selectric printer is unmatched in quality. Prices for such a printer on the used market are also very attractive. These printers are old (but fully repairable by typewriter stores), slow, and above all, not reliable. (The type-ball positioning bands will not take sustained high-speed output and go out of adjustment or break.) Selectric-type printing terminals are good enough to make offset masters for printing, offer a wide range of typefaces, and provide you with a good typewriter when your computer isn't using it.

Most daisy-wheel printers are from Diablo, the company that made this printing technique practical; Qume, a company formed by former employees of Diablo; and NEC, a Japanese company. (The NEC printer uses a "type basket" instead of a daisy wheel, but otherwise the printer is about the same.) Radio Shack also sells an imported daisy-wheel printer similar in construction to the Diablo-type printer.

The printing element of daisy-wheel printers is a spoked wheel with a type pallet molded on the tip of each spoke. The selected character is spun into position where a hammer strikes it. Until recently (the last couple years or so) the daisy-wheel printer was the only printer offering dot-addressable graphics and proportional spacing. The speed of these printers is typically 30 to 60 characters per second. While not high-speed printers, they produce typewriter-quality printing and take any kind of paper you can put into a typewriter. The images printed are usually not as clean or dense as those of a Selectric printer, but the mechanism is much more reliable and at least twice as fast.

The dot-matrix printer was originally designed to print lottery and racing tickets back in the early 1960s. These designs were meant to be used in very light duty, low-speed applications. Gradually, print-head improvements provided faster needle operation and longer lasting heads. The microprocessor came along in the late 1970s, and soon there was a huge demand for low-cost printers for low-cost microcomputers. Without the microcomputer, the demand for printers of this type would not support the present market, and without the microprocessor, today's dot-matrix printers could not be made, an interesting case of symbiosis.

The dot-matrix printer is the most widely accepted printer technology on the market, because it can be mass produced cheaply for low-performance, basic printing applications, yet with more sophisticated design can print much faster, with high quality. Dot-matrix printers still cannot produce typewriter-quality images, but are doing an increasingly good job at trying. The dot-matrix printer can electronically change character fonts, automatically justify text, provide intercharacter proportional spacing, print in four or more colors (with overstrike), automatically feed letterhead paper, and they can do this fast. Even the slowest popular models such as the Epson MX-80 or MX-100 have throughput rates over 30 characters per second.

What to Ask the Salesman

Since nearly every manufacturer has a different way of saying the same thing, and there is a lot of "specsmanship" involved, talking to a salesman can be confusing. (Specsmanship is where specifications are written so the reader thinks things are better than they are.) Here are a few terms and their definitions that are bandied about. You should understand them before you begin to compare printers.

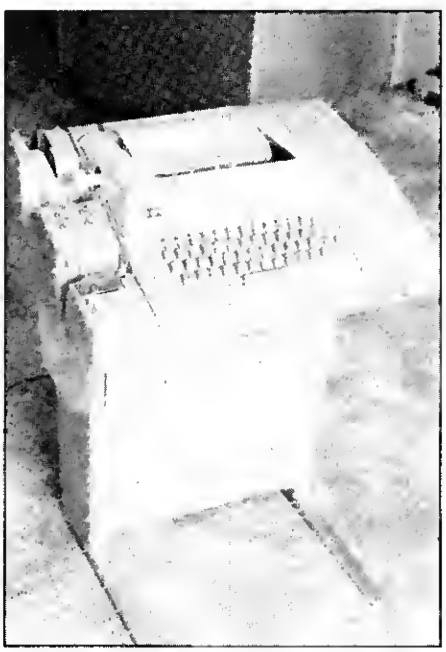


Photo 2. The ubiquitous Teletype Model 33. Probably the lowest-cost terminal/printer you can own. This printer is a 10-character-per-second unit, uppercase only. Features include a paper-tape reader, punch, and keyboard. These printers were designed in the 1950s and over a half million of them were produced. They are available on the surplus market at around $250, and will last forever if properly maintained (oiled) and you keep the mice out of them.

Cps usually means characters per second. Advertising might say that a given printer is an 80-cps printer. What they actually mean is that it is possible for the printer to print something at a rate of 80 characters per second. This might be two adjacent periods. In any event, this specification is meaningless.

The only meaningful speed rating is throughput. This is the number of characters or lines of text that is actually put down on paper in a given unit of time. You can measure the throughput of a given printer with a simple program that outputs a given number of characters; simply divide the number of characters by the time it takes to print them. This figure varies with the length of the line and other printer features, but in most low-cost printers, it is something around 30 to 40 characters per second.

Near letter quality (nlq), correspondence quality, and other such terms are the way manufacturers say that they have a print head that provides overlapping dots. This closes in the dot pattern and makes the characters look more fully formed. Some printers overlap dots only vertically, others both vertically and horizontally. You will probably find that those printers with nlq have a shorter (typically .09 in.) character, rather than the standard .1-inch-high matrix character. Unless you prefer larger characters for easy reading, there is no reason to choose a printer on this issue alone.

Lowercase descenders: Are they above or below the base line? It is cheaper to build a printer with a seven-needle head than a nine-needle version. If the print head has only seven needles, then lowercase characters must be printed entirely above the base line. One of the most popular terminals I've used is the Decwriter II from Digital Equipment Corporation. It has only seven needles. The lowercase g, j, p, q, and y characters are scrunched above the base line. No legibility problem, but I recommend a printer with nine needles if everything else is equal, or if print quality is important.

Printer buffer: This is the internal memory capacity of a printer. Cheaper printers buffer only one line. What this means is that the printer can gobble up only one line of text at a time from your computer, print that line, then gobble another. In the meantime, you and your

computer must sit around and wait for the printer. Larger printer buffers, some as big as 3,600 bytes (or characters), allow the printer to consume that much text, then free your computer to do other things while it prints the buffered information. Special programs called spoolers do much the same thing.

Character sets and fonts: A character set is the list of all printable characters. Typically, this is all the ASCII characters from CHR$(32)—a space—to CHR$(127). A printer that prints all these characters is called a 128-character printer. A 64-character printer is one that only prints the capital letters A through Z. If you are interested in a printer like this, be sure that it can be set to print the uppercase equivalents of the lowercase characters. (Sooner or later you will send lowercase text to it.) A font is the design of the characters in a character set. For example, the two most common typewriter fonts are pica and elite. Spec sheets that do not mention fonts mean the printer has only one font.

A printer that can be loaded with a font from your computer is said to have "downloadable" fonts. This may seem like a powerful feature, but a complete character set is difficult to design. (I have done it, and it takes several weeks of fulltime work to just set up the character-set dots on paper.) Downloadable-type printers sometimes don't allow you to replace just a single character. Further, many printers of this variety don't have the high density, overlapping dot fonts of the near letter-quality printers.

Multispeed printing: Manufacturers have discovered that by printing fewer dots and spreading them apart, they can make a printer with higher throughput. Since the character-set font has fewer dots, the quality of it is not as good, and so these high-speed fonts have a variety of names. So-called "draft" fonts can nearly double the throughput of a given printer.

Proportional spacing is the ability of a printer to vary the amount of spacing between characters. Intercharacter proportional spacing allows a printer to print text so that it can be justified; the right margin of the text can be made square like the left, without simply doubling up on the spaces between words. Most daisy-wheel printers have this capability as does the Centronics 739 printer. In these cases, your computer must maintain a table of character widths and tell the printer how much to space each character. Integral Data Systems IDS 460, 560 and the entire Prism and Microprism line feature automatic proportional spacing.

Text justification, as mentioned earlier, is the squaring off the right margin of print. If the printer can proportionally space characters, then by keeping track of the amount of print space left on a line and the length of the following word to be printed, you can add or delete space between characters so the line is completely filled each time. You naturally do not want to break a word in the middle, so each line must be examined and the spacing computed before it is sent to the printer. A very few printers provide automatic text justification. When enabled, text is justified within the bounds set by the margins. Margins can be set to any width so that newspaper-like columns or full-page lines alike can be automatically filled with text, without any assistance or text processing from your computer.

Most dot-matrix printers in the low price range have a number of miscellaneous print features that add to their

versatility. Bolding, available on some, is simply an automatic overstrike. (The printer prints a bolded line twice.) Double-width characters are printed by doubling up on the horizontal dots, making characters twice as wide.

Many printers have selectable pitch. Names for pitch vary. Epson talks about enlarged or condensed characters, while most American companies use a character-per-inch (cpi) term. Standard pitches are 10, 12, and about 16 characters per inch. (The 16-character-per-inch pitch is important since it allows a 132-character line to be printed in eight inches, and therefore on 8½-by-11 pages.)

Underscore capability is difficult for low-cost printers because a line through any dot-matrix character just about destroys legibility. Dot-matrix printers have met the challenge three ways. The first is to ignore the issue and let the user manually underscore text using a period or graphics. Second, some printers have a special font that leaves the bottom (ninth) dot free. Automatic underscore simply fires the bottom needle for every character. The third variation is to provide underscore on the ninth dot and wipe out the lowercase descenders. (This makes it difficult to read words with descenders in them, but does give something to print on the spec sheet.)

At this writing Centronics and Integral Data Systems are the only companies selling a low-cost color printer. The Centronics 739 printer can be fitted with an option for a red-black ribbon. The IDS Prism printers use a four-color ribbon. Colors can be mixed by overstrike.

Graphics capability is a natural for dot-matrix printers. Instead of internally looking up a list of dots to be printed from the ASCII character code, the only difference in printing graphics is that the ASCII character code itself is printed. Printers differ in the way the character codes are handled. For instance, Centronics uses graphics characters only above 32; this allows control codes such as carriage returns and line feeds to work normally. IDS uses all ASCII values from 0–127, using a control code as an escape from graphics. Epson requires that the computer tell the printer how many graphics characters are coming, then prints all the following characters as graphics. The character codes themselves are printed in two ways. Bit zero of the ASCII code can be used to print either the top or bottom needle of the head. Each following bit of the ASCII code is used to control the following needle.

The printer's resolution is about the only feature mentioned in the graphics part of a specification sheet. Horizontal resolution is determined by how fast the head moves across the paper and how fast the needles fire. This is typically over 72 dots per inch, but can vary. Vertical resolution is controlled mechanically by needle spacing in the head. The only way this can be changed is to change the head. Generally, the more dots per inch, the finer the printed image. Most graphics output comes from screen dumps, and the resolution of these graphics is limited to that of the screen, regardless of your printer.

Mathematically, if the vertical and horizontal resolutions are the same, images printed will not be distorted. If a circle is plotted, then sent to a printer with 100-by-100-dot-per-inch resolution, the circle prints round. If the printer had a resolution of 100 by 80, the circle becomes egg shaped by about 20 percent. If your printer has a resolution near 80 dots per inch, it is state of the art. If the image distortion due to

nonsymmetrical resolution is important, it can be corrected by software.

Your printer's interface is important. There are two kinds: parallel and serial. A parallel interface is common among the cheaper printers because it requires less electronics and is simpler to control. Basically, a parallel interface requires a common ground, a signal for each bit in the characters to be sent (normally 7), a strobe (the signal from your computer that says a character is ready for the printer to pick up), and a busy or acknowledge from the printer that says it is ready for the next character.

If the printer doesn't want another character, it signals "busy" or doesn't acknowledge the last character sent. Your computer waits to send the following character until the busy or acknowledge is returned, and that is why it hangs up if your printer if offline when you try to print to it. The so-called "Centronics" parallel interface (which, incidentally, has no formal standard) is the most common parallel interface, and usually printers and computers with this interface work together without any problems.

Serial interfaces are required if you want to use your printer with a terminal connected with timesharing computers. Many low-cost printers do not offer a serial interface, or offer it as an extra-cost option. Serial data requires a ground, a data line from your computer, and a signal to tell your computer when to stop. This may be the Data Terminal Ready (DTR) signal that is either on or off, or an actual ASCII code for control S (XOFF) or control Q (XON). The XON-XOFF code is used when you are connected through telephone lines. Data Terminal Ready can only be used if your printer is "hardwired" to your computer.

The problem that printers have with serial data is that the data can come in just about anytime. Also, it has to be serviced within one character time, or the next character can arrive and be lost. If you are running serial data at 9,600 baud, for example, the characters have to be handled in 1/960th of a second.

Another problem is that when the printer wants data to be stopped, there may be another character already on the way, or when XON-XOFF is used, 10 or 20 more characters may arrive before the computer gets around to stopping them. The management of serial data requires processing time from your printer, plus a buffer area in memory to handle the data. Many low-cost printers do not have the software sophistication or memory to handle serial data, and so the serial option becomes a little box that lives outside the printer.

I can't think of any reason why one interface should be preferred over another, if you can run serial at 9,600 baud. Serial interfaces might be considered more flexible since you can use them with a terminal and a computer. One thing that is not important is the parallel data-transfer rate. I once found a Japanese firm extolling the parallel data rate—how fast characters could be accepted by the printer. This specification has no meaning as long as it is faster than the throughput of the printer.

Paper-handling ability is an important aspect of the printer. Until recently (the last year or so), most printers required perforated continuous stock, because it is difficult to design a printer that will take a single sheet of paper or continuous stock equally well, and because the print quality of earlier dot-matrix printers was so poor that nobody wanted to send a letter printed by them.

"Educate yourself to be coldly objective about what you see, hear, and read."

With the advent of the improved character sets, all this has changed. Low-cost printers can now take single-sheet paper, but for the most part, paper insertion is difficult and you must carefully align each sheet for left and right margins as well as for the top edge. I am not aware of any low-cost printer than can take an envelope, but this problem will probably be solved within the next two or three years.

The operating costs of a printer are primarily a function of ribbon cost in terms of dollars per characters printable, and the cost of a replacement head. Ribbon lifetimes can vary, depending on how fast they are used (the ink can dry up over a period of time) and how fussy you are. Ribbon usefulness is subjective; a dead ribbon for one person is only half gone for someone else. You had better believe that the ribbon lifetime on a specification sheet means the end of lifetime to everyone.

Quoted print-head lifetime is a statistical matter. Print heads can die in several ways. The jewel in the nose of heads (it is a bearing that aligns and guides the print wires to the paper) can crack or wear out so the wires are not held evenly. This results in "wavy" characters. It can also gum up with ink and paper lint. This results in a sticky needle that won't work well when it is cold. Other electrical or mechanical faults usually stop operation of a needle. The lifetime of a head is usually specified in millions of characters. Cheap, disposable heads last 30 or 40 million characters. Nondisposable heads can last 200 or 300 million characters. Compare the cost on a character-to-dollar basis to find which is the best bargain. (The more expensive heads usually win.)

Off to the Market

Now, how does one go about shopping for a printer? Unless you live near a good-sized town that has several computer stores, you are going to have to trust in the reviews that you have read in magazines or pick a printer that a friend has, simply because it is the only one you have seen work.

First, educate yourself to be coldly objective about what you see, hear, and read. Second, slow down and take the necessary time to find your printer. This can not only save you money, but also get the printer that fits *all* your needs.

Let's practice by picking a printer for someone that would have a typical variety of printing needs. Assume that you are a college student in the junior or senior year. By this time, a student would have a variety of text that could be edited and printed by a small computer system.

If this student were involved with mathematics or statistics, then the computer would certainly be an asset to him in other areas, especially if the printer could handle graphic output. An economics major might find one of the popular spread-sheet programs of use. These requirements are not too far removed from those of a small business user, except that small businesses would generally run the same programs every month, and usage might require a heavier-duty printer. Since small businesses might use the system to run billing or other higher-volume data through the printer, throughput is a greater concern.

So let's pick a general-purpose printer in the low price range. This printer must print near-letter-quality text, because you will want to print term papers and the like on it. Since term papers typically use single-sheet paper, it must be able to print on typewriter paper as well. The printer should be able to super and subscript text, as well as underline and overstrike text for bolding.

Already, the list of specifications has gotten us into trouble. There are only one or two printers on the market that can automatically fill all the above needs, unless you go shopping for a word processor that supports your printer. You might not only have to look for a printer in your price range, but will have to select and pay for a word-processor package as well.

The next step is to get a collection of magazines together and systematically compile a list of available printers, phone numbers, and addresses of each manufacturer. Call those with 800 numbers and write (or circle bingo numbers) all the others that interest you. You may find that literature reaches you faster if you write directly rather than using bingo numbers (reader service cards) offered by some magazines.

Check the literature off the list as you receive it. Don't start your decision-making process until you have all your data, unless a given printer sells for more than your budget. Simply mark your list as "price out of bounds," but keep track of all printers until you have made your final decision.

Make a chart with columns for brand name, model, and each feature that you want. A student will want proportional spacing, a near-letter-quality font, super/subscripting, single-sheet paper handling, probably graphics, and price. You might also want some idea of the throughput of a given printer, but you will probably have to find this in a review since advertising literature normally mentions only the character-per-second print rate. The last column should be the retail price. Use this instead of any mail-order prices you might have found. (Comparing mail-order prices may be misleading.)

Ignoring any gloss that you might find on the brochures, examine the claimed performance of every printer and check off the columns for each feature that is important to you found on the printer. Ignore any features (for now) that you didn't put on the chart.

After you've entered the entire stack of printer literature onto your chart, set aside the literature for the printers that do not meet your needs, and don't let them influence any further product selection.

Your next step is to read product reviews for the printers that made the first cut. Hopefully your library has a couple computer magazines. If not, order back issues of *80 Micro* and other magazines that have a cumulative index. From that you can find the issue that has the review you are looking for.

A book that you may find useful is called *The Index* by W. H. Wallace. (Missouri Indexing Inc., P.O. Box 301, St. Ann, MO 63074, 314-997-6470). This book is an index of over 40 magazines printed to about mid-summer, 1981, most from the first issue.

The important thing about reading reviews is to try to get a feel for the author. I don't know how many times I've read something like, "This is the first widget I've ever used, and it is the best one in the world." The best thing you can do with a review of this caliber is to trash it. Look for an objective review, keeping in mind that there is something wrong with every product, and if the reviewer doesn't mention something bad, he hasn't done a good job.

At this time you should also visit computer stores in your area, and friends (and strangers) that have printers of the type you are considering. After researching reviews, your list of possibilities should narrow. Now you must assess the product support for the remaining printers. Where do you get the printer fixed and how long will it take? If you call the factory (or importer), can they give a competent answer to your questions? Is there any "canned" software that is necessary for your application?

"Don't spend your money until you understand what you are buying."

Typically, there are three possible places to get your printer fixed. The most obvious is to take it to the place you bought it. Results of this type of service vary, but if you have purchased products from a computer store and are known there, your service is likely to be much better than if you buy your printer by mail order. Part of the higher price of retail sales is for customer support, to be rendered before and after the sale. This hand holding, as it is sometimes called, doesn't come free. Mail-order houses can laugh at you if you have a problem.

In any event, check into factory-authorized service facilities; how many are there, where are they located, and so on. You can find this information in the literature you already have. If not, call the printer company and ask. Don't let yourself be talked into the idea that the printer is so reliable it doesn't break. That is pure balderdash. Ask about company factory repair, and the usual turnaround. If you do a thorough job, you will also call the factory service center and ask, "My model XYZ just broke (nothing happens when I turn it on, and there is a strange smell inside). Any guess how long and how much it will take to get it back?"

Many import products have little or no service program and rely on random dealers to fix broken printers. Since the dealers are very often undersold by mail-order houses, they are not terribly interested in fixing the mail-order specials, and so you suffer the results. The other side of this is that dealers very often have not been trained by the importers and have to rely on manuals not much different from user manuals sent with the product. In the search for your printer, place a heavy value on the quality of service you can expect when it breaks.

Not many importers offer help over the phone; American companies do better in this respect.

The final consideration in product support is perennial for both computers and peripherals. What software is available for the product? Word processors are important, but many applications do not need them. Many newer features such as proportional spacing built into the printer cannot be used on popular word processors. Graphics is another area where existing software is vital. Most people either could not or will not develop a graphics handler for their printer. If graphics is important, and you want to do more than a simple screen dump, look for a graphics package to drive the printer before you buy it.

If you have conscientiously followed the guidelines I have given you, the printer you buy will suddenly pop out of the list. I can't tell you which one is best. This decision is one that you must make. But you can make a wise decision if you take the time to learn the jargon around printers and examine everything you can from advertising, product reviews, trade journals, retail stores, and friends.

The important thing is not to hurry. Don't spend your money until you understand what you are buying. And above all, don't give up and just buy the printer that is most widely advertised. When you do that, the advertisers win, and you might come in second place. ■

Jim Hansen's address is P.O. Box 234, New Boston, NH 03070.

Suppress Those Demon Transients

by G. Michael Vose

Your micro needs good, clean electrical power to run at its best. Here's what you can do to minimize noise and voltage disturbances.

Like a dark and slimy creature living in the bowels of a nightmarish fantasy, there is a demon hiding in the nooks and crannies of your computer room. These demons force themselves into the lifeblood of your computer system—its electrical power lines—to zap your data or crash your programs.

These creatures have supercharged names—spikes, surges, transients, sags, pulses—and can infect any electrical power-transmission grid. They are impossible to see, difficult to measure, but fortunately, are easy to eradicate.

Understanding the dynamics of power-line disruptions will help you to cope with these demons in your computer room.

Dirty Power

Electrical power can be suspect for a variety of reasons. Power fluctuations can be transmitted along utility power lines, or they can be generated within a building or room by a number of sources. The kinds of electrical interference that can affect computer devices include:

- Transient voltage above or below the normal voltage for the system (called spikes, surges, dips, sags, or brownouts).
- Noise interference (often called hash by electrical engineers).

According to studies by the Institute of Electrical and Electronic Engineers (IEEE), harmful power line disturbances occur as frequently as four times a day on average. Of these disturbances, 88 percent are power spikes (transients and surges), 11 percent are brownouts (low voltages), and .5 percent are power outages.

The difference between transients and surges is strictly one of duration and degree. A transient is a voltage increase to 6,000 volts or higher lasting 100 microseconds or less. A surge usually lasts slightly longer but is characterized by voltages of 3,000 volts or less. Voltage surges of 1,000 volts are common, everyday occurrences in most industrial and commercial electrical environments.

The most common causes of electrical interference are voltage imbalances generated by the operation of electrical machinery. A refrigerator motor or a furnace blower motor can cause voltage spikes and voltage reductions during operation. Other causes of interference include the obvious, like lightning, and the not so obvious such as sunspots.

Spikes and Sags

What kind of damage is caused by increased or decreased voltage? Commonly, little permanent damage occurs to a computer as a result of most voltage fluctuation. It is possible in the case of an extraordinary event such as a lightning strike for sensitive semiconductor devices, such as memory chips and microprocessors, to experience circuit melting, corrosion, arcing (which will break down insulation), and

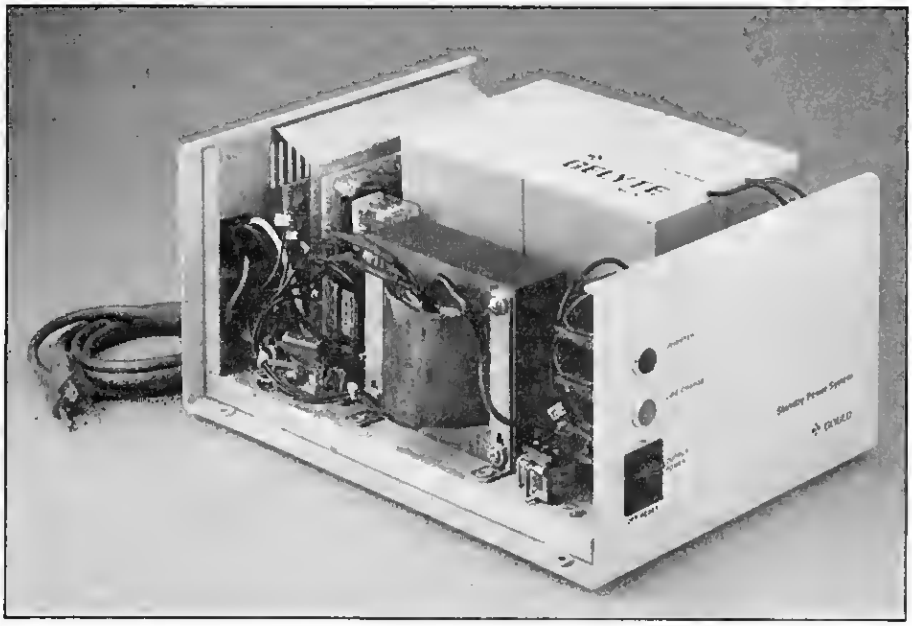


Photo 1. The Gould Standby Power System

reduced component life. Disk drives and other mechanical components can suffer pitting and metal transfer amongst switch contacts.

The most common effect of voltage transients on computing devices, however, is alteration of memory. Small voltage fluctuations can be interpreted by low-power computer circuitry as legitimate input signals. Therefore, random memory errors are often a direct result of electrical interference.

Semiconductor memory chips contain circuitry that comprises hundreds of individual memory cells. These cells are composed of field-effect transistors that act as gates that can either be open or closed, coupled with capacitors that store an electrical charge.

During a read/write operation, the transistor that acts as a gate is opened and a small electrical charge (for a binary 1) or no charge (for a binary 0) is stored in the accompanying memory cell capacitor. If the data being written is subject to a spike or surge during the read/write operation, a memory capacitor could be set high instead of low. Since memory pulses are stored simultaneously one bit per chip over eight chips in each 16K of memory, an entire byte of memory could be written incorrectly. If this happened to be a vital byte, an entire data block would be rendered unusable.

Equally critical is the effect of overvoltages on disk-drive units. Since the read/write head in a floppy disk drive is designed to alter magnetic fields, voltage spikes often cause the head to alter the data on a disk even when the head is engaged in a read operation. These alterations will occur randomly. However, since the disk reads the directory tracks more than any other track on a disk, the probability that a random glitch will alter the directory is high. When this occurs, the entire disk can be rendered unusable.

Voltage spikes are most likely to occur as the result of the switching on or off of an electric motor. Copy machines, elevators, air conditioners, printing presses, furnaces, and so on all generate voltage spikes. These spikes occur within the building where the equipment is located and are transmitted to all parts of the building along its electrical wiring.

Low voltage, commonly called a dip or sag and resulting in a 5–20 percent voltage reduction of approximately a second in duration, often occurs when there is a sudden call for power by an electrical device. Low voltages can cause the picture of a television or video monitor to shrink visibly for a brief moment. Commonly, low-voltage periods are followed immediately by a high-voltage surge as the system draws power from the outside to satisfy the needs of its consuming devices.

Photo 2. J. W. Miller Line Filters

Brownouts are long-term voltage reductions, typically of 5–8 percent. They usually are gradual in nature and may last from several minutes to several hours. Brownouts are initiated by the utility supplying the power, usually due to equipment failure or excessive demand on the power grid. Because they occur gradually, brownouts seldom cause difficulty in computers since the power transformer in the computer adjusts the incoming power to suit the needs of the system. These transformers are ideally suited to accommodating gradual power fluctuations.

Noise Interference

Voltage fluctuations are the more serious of the two kinds of electrical interference, but noise interference comes in a greater variety of forms. Noise is simply an unwanted electrical signal and can be as unobtrusive as a jolt of static electricity. Anyone who has ever worked in a computer room without antistatic carpet knows that static electricity can cause data loss in computers.

The common forms of noise are:

- Radio frequency interference (RFI) caused by the reception of electromagnetic signals (radio/tv transmission) by power lines acting like antennas.
- Electromagnetic interference (EMI) caused by a fluctuating magnetic field such as that produced by an operating electric motor.
- Electromagnetic pulse (EMP) caused by a collapsing magnetic field such as lightning. Static electricity is an EMP.

Noise occurs at lower magnitudes than voltage spikes and is less dangerous to equipment. However, it can degrade performance and cause data loss in computers, because noise occurs at extremely high frequencies and high speeds. The response time for most power transformers is not sufficient to trap these high-speed signals.

Noise can be generated by many common devices: fluorescent lights, radio and television transmitters, electric typewriters, hand tools, automobile ignition systems, and sunspots. Computers themselves generate some RFI and their disk-drive motors and power supplies generate EMI. Loose electrical connections and damaged power cords are other sources of noise.

What Can You Do About Spikes and Noise?

The obvious solution to the problem of "dirty" power lines is to incorporate into the distribution system some sort of filter. In much the same way that an oil filter extracts impurities from your car engine's motor oil, a power-line filter can knock down voltage transients and eliminate noise. This is done using circuitry specially designed to handle the offending voltage fluctuations.

Power-line filters come in a variety of types, sizes, and voltage configurations. The cost spectrum for these devices ranges from $20 to $2,000. Some power-line filters merely dissipate voltage spikes; others offer battery backup to guard against power sags, brownouts, and outages, and one system even prints out a record of every occurrence of a power-line fluctuation.

Table 1 shows a representative cross section of the devices for power-line filtering currently on the market.

Typically, power-line filters work by rerouting unwanted voltage through circuitry equipped with electrical components called varisters that cause the excess electrical energy to be given off as heat. In some cases, part of this excess energy may be diverted to the main electrical system's grounding loop.

Power-line filters should be installed between the main source of power and the device to be protected. Normally the power-line filter plugs into a wall receptacle and then your computer plugs into the filter. Many filters have an on/off switch so that the entire circuit can be turned off when your machinery is not in use.

The principle qualities to look for in a power-line filter are the overvoltage-handling capacity, the response time, and the frequency range for noise suppression. Usually, a low-cost filter offers only minimal protection. In fact, the lower-priced filters may offer overvoltage protection only with no noise-suppression capability. Furthermore, only the most expensive filters offer battery back-up for protection in the event of brownouts or power outages.

The overvoltage-handling capacity of power-line filters ranges from 1,000 volts to 12,000 volts. A capacity of 5,000 volts will protect the line from any spike other than a lightning strike. Many manufacturers call this capability the "clamping voltage." This refers to the maximum voltage that will be "clamped," or suppressed by the device.

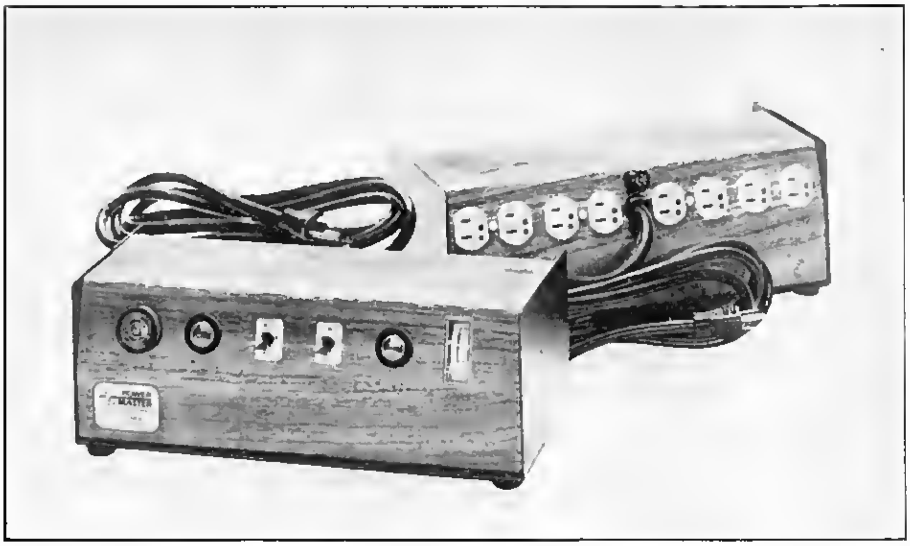


Photo 3. The Power Master Line Monitor from SGL Waber Electric

The response time of power-line filters typically ranges from 5 to 15 nanoseconds (1 nanosecond is one billionth of a second). Noise-reduction frequency response is usually in the range 150 KHz to 30 MHz. Some devices are rated as high as 200 MHz. Unless your building is near a microwave transmission site such as a tv station or a radar installation, an upper level of 20 to 30 MHz of protection should be sufficient.

Picking the Right Filter

The line filter that you choose for your particular application will depend on several factors. Principle among these is the investment you've made in hardware and software. Second, you must consider the cost of having your system crash. In a business setting, a lost inventory file could require dozens of man hours of labor to rebuild, which could mean several hundred dollars in direct expenditure plus lost sales or decreased efficiency.

You must also consider the frequency and severity of electrical interference in your operating environment. The condition of the electrical wiring in your building and the number of electrical devices drawing power from the system are important factors. Your power company might be able to produce statistics on the frequency of voltage irregularities in their service area. The electrical specifications of the power supply in your computer will reveal the degree of its susceptibility to electrical interference—the manufacturer can usually provide this information.

Finally, you will want to consider the aesthetics of the variety of units on the market. Some devices plug directly into a wall socket and are unobtrusive, while others feature multiple-socket design arranged in a so-called "power strip." Yet others are housed in boxes of varying size and color that become part of your computer installation.

Ultimately, you may decide that your computer functions well enough in its present environment. The extra expense of another piece of hardware may be unnecessary. On the other hand, if your machine seems possessed by demons that randomly chew up data or spit out garbage, you may want to consider cleaning up your electricity supply. Sending nasty notes to the power company is optional. ■

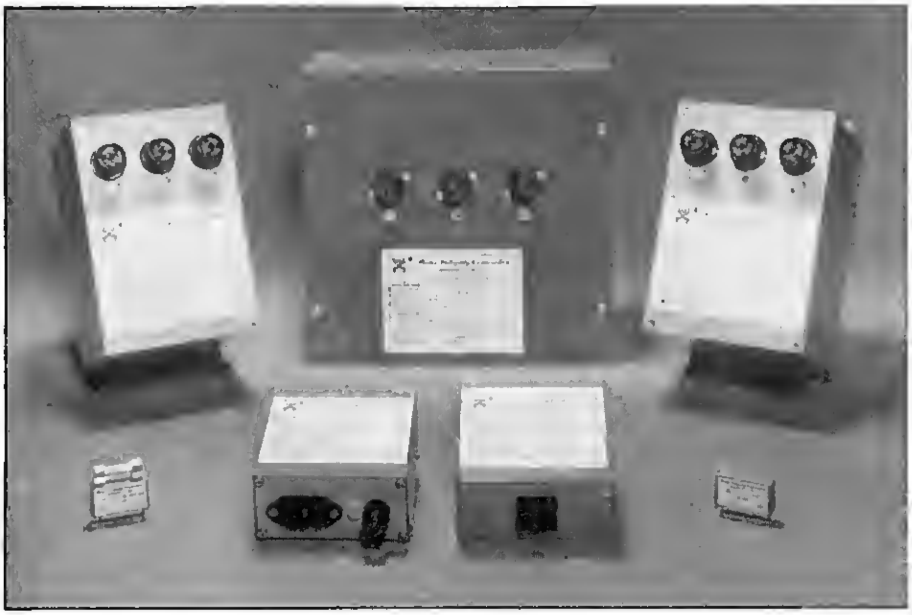


Photo 4. Power Integrity Corporation's Transient/Surge Suppressor Line

G. Michael Vose is a technical editor for 80 Micro.

1. Voltector and Voltector Multibus Strip
$79.50
Pilgrim Electric Co.
29 Cain Drive
Plainview, NY 11803

2. Super Isolator
$95
Electronic Specialists, Inc.
171 Main St.
Natick, MA 01760

3. Genisco C Filter
Genisco Technology Corp.
18435 Susana Road
Rancho Dominquez, CA 90221

4. Multi-Outlet Power Strips
$43.97–$73.95
MFJ Enterprises
PO Box 494
Mississippi State, MS 39762

5. Spike-Spiker Mini-I and Mini-II
Kalglo Electronics Co.
6584 Ruch Road
East Allen Township
Bethlehem, PA 18027

6. Isolator (Iso-1, 2, 3, 4, and Iso II)
$62.95–$106.95
Electronic Specialists, Inc.
171 Main St.
Natick, MA 01760

7. MayDay
$240 and up
Sun Research, Inc.
Box 210
New Durham, NH 03855

8. Lemon, Lime, Orange, Plum
$49.50–$139.95
Electronic Protection Devices
41 Sun St.
Waltham, MA 02154

9. Power Master Line Filters
SGL Waber Electric
Dept. W
300 Harvard Ave.
Westville, NJ 08093

10. HDA Power Master
Automated Systems, Inc.
5265 Port Royal Road
Springfield, VA 22151

11. EFI Filter/Suppressor
$400–$800
Electrical Filters, Inc.
PO Box 9087
2147 East 3300 South
Salt Lake City, UT 84109

12. Clipstrip and Clipper
Line Voltage Transient Suppressors
DyMarc Industries, Inc.
7133 Rutherford Road
Baltimore, MD 21207

13. The Voltage Surge and
Transient Suppressor
Cuesta Systems, Inc.
3440 Roberto Court
San Luis Obispo, CA 93401

14. LF2 and LF6
R.L. Drake Co.
540 Richard Street
Mimisburg, OH 45342

15. Power Line Filter (26-1451)
$49
Radio Shack
(All Radio Shack stores)
Ft. Worth, TX 76102

16. RAMLOK
Ladco Development Co.
PO Box 464
Olean, NY 14760

17. ZTA and ZTD Filters
Power Integrity Corporation
300 East Wendover Ave.
Suite 102
PO Box 9682 Plaza Station
Greensboro, NC 27401

18. Quiet Line 6
$39.95
BWJ Technology
Box 6214 Dept. 8
Arlington, TX 76011

19. Glitch Guard POP
TII Electronics Division
1375 Akron St.
Copiague, NY 11726

20. Glitch Sentinel (GS-1, GS-2, GS-3)
$300–$900
Billings McEachern, Inc.
333 Cobalt Way, Suite 106
Sunnyvale, CA 94086

21. Standby Power System (SPS)
$489
Gould, Inc.
PO Box 43140
St. Paul, MN 55164

22. Power Line Filter
J.W. Miller Division
Bell Industries
19070 Reyes Ave.
Compton, CA 90221

23. Model CN-1110
DoAll Company
Des Plaines, IL 60016

Table 1. Power line filtering equipment

Into the 232

by Howard Miller

Ever wonder about the RS-232 port on the back of your micro? Just how does it receive and send data? Stay tuned. Howard's got the scoop.

The TRS-80's serial interface receives parallel bytes of information and sends them out one bit at a time. This makes data transmission between machines simpler; only a single pair of lines is needed to transmit signals from one point to another.

The factors involved in sending information this way, such as which bit (most significant or least significant) is sent first, how the start and end of a character are defined, and the current flow in the line, are all decided in advance. The hardware details, determined a number of years ago, are collectively referred to as the RS-232C standard.

The RS-232C standard allows information to be sent from one machine to another even though the machines were made at different times and by different manufacturers. There are actually 25 lines in the standard and 25 pins in an RS-232C connector, but it is not necessary to use all of them.

Pin Number	Abbreviation	Description
1*	—	Ground
2*	TXD	Transmitted data
3*	RXD	Received data
4*	RTS	Request to send
5*	CTS	Clear to send
6*	DSR	Data set ready
7*	—	Signal ground
8*	CD	Carrier detect
9	—	Used for data-set testing
10	—	Used for data-set testing
11	—	Unassigned
12	SCF	Secondary carrier detect
13	SCB	Secondary CTS
14	SBA	Secondary TXD
15	—	Synchronous transmit clock
16	SBB	Secondary RXD
17	—	Synchronous receive clock
18	—	Unassigned
19	—	Secondary RTS
20*	DTR	Data terminal ready
21	—	Signal quality detector
22*	RI	Ring indicator
23	—	Synchronous clock rate
24	—	Transmit signal timing
25	—	Unassigned

*Lines used by TRS-80

Table 1. RS-232C Pin Designations

Current, rather than voltage, levels are used to transmit information. In this way, the resistance of the lines is extremely low and interference from outside sources such as lightning, fluorescent lights or radio is reduced. There are actually two data communication channels in the standard, but the serial interface of the TRS-80 only implements one of them.

RS-232C Lines

The transmit data line (TD) connects the serial output of the interface to whatever is on the other end. I will call the TRS-80 and its serial interface the data terminal and whatever is on the other end the data set. Receive data (RD) transfers information from the data set to the data terminal.

DTR tells the data set that the data terminal is ready and DSR tells the data terminal that the data set is ready. Request to send (RTS) lets the data set know that the data terminal has information it wishes to pass to the data set. Clear to send (CTS) lets the data terminal know that the data set is ready to accept information.

Two more lines are used by modems to give additional status information to the data terminal. These lines are carrier detect (CD) and ring indicator (RI). Carrier detect signals the data terminal that the modem has detected the mark frequency on the telephone line or other communication channel it is attached to. This lets the data terminal know it is in touch with another data set. Ring indicator lets the data terminal know that the telephone is ringing; on more sophisticated systems, the data terminal can answer the phone.

Table 1 shows the pin assignments of the RS-232C standard.

UART and ASCII

The heart of the TRS-80 serial interface is the large-scale integration (LSI) chip known as the universal asynchro-

nous receiver transmitter (UART).

The particular UART used in the TRS-80 is a TR1602B, but several other chips are pin-for-pin compatible with it.

The TR1602B has two tri-state data buses and some tri-state status lines. Input lines get data into and out of the UART, enable the status lines, and make it possible to modify the way the UART operates. Tri-state lines simplify the design of the serial interface by electrically isolating various outputs from a common line until they are called upon. That way, several outputs can be tied to the same line without shorting anything out.

"...for exchanges between computer bulletin boards or remote data bases, seven bits are usually used."

The serial output line (SO) of the UART maintains a logic one, or mark condition, whenever it is not actively engaged in sending out information. When data is ready for transmission, the SO goes to logic-zero condition (space) for a predetermined period of time. This mark condition is referred to as a start bit, and is immediately followed by the bits that make up the data sent to the UART by the computer. Now, the computer normally uses data made up of eight bits (a byte), but the number of data bits that the UART sends out isn't necessarily eight. In the TRS-80 serial interface, the number can be five, six, seven or eight—more about that later.

Immediately after the data bits, an optional parity bit can be sent. Its purpose is to indicate the number of ones and zeros (marks and spaces) in the character just sent. This allows the data terminal at the other end to determine if the information has been garbled in transmission. Many services, such as computer bulletin boards, omit the parity bit.

After all this information has been sent, the UART returns to the mark condition for a specific period of time to let the data terminal at the other end know it has finished sending that particular bit of data.

Data can be sent in the 8-bit format, but for exchanges between computer bulletin boards or remote data bases, seven bits are usually used. In this format letters and numbers are transmitted; special codes allow control of the data terminal from the other end. The letters, numbers and codes are usually in the format known as the American Standard Code for Information Interchange (ASCII). In this format the number of stop bits can be either one or two. Another code format known as Baudot uses only five bits in sending letters, numbers and control codes. Baudot code requires 1½ stop bits, which the UART automatically provides whenever it is set up to send 5-bit code.

Input Port

D7—Receive data available
D6—Transmit buffer empty
D5—Receiver over run
D4—Receiver framing error
D3—Receiver parity error
D2—Not used
D1—Not used
D0—Not used

Output Port

D7—Parity odd/even
D6—Word select bit 1
D5—Word select bit 2
D4—Number of stop bits
D3—Parity enable
D2—Serial data out direct from computer
D1—Request to send
D0—Data terminal ready

Table 2. I/O Port 234 Configuration

Input Port
(DIP switch configuration)

D7—Parity odd/even
D6—Word length bit 1
D5—Word length bit 2
D4—Number of stop bits
D3—Parity enable
D2—Baud rate bit 2
D1—Baud rate bit 1
D0—Baud rate bit 0

Output Port
(Baud Rate Generator)

D7—Transmit clock		bit 3
D6—"	"	bit 2
D5—"	"	bit 1
D4—"	"	bit 0
D3—Receive clock		bit 3
D2—"	"	bit 2
D1—"	"	bit 1
D0—"	"	bit 0

Table 3. I/O Port 233 Configuration

Even though the UART is said to send and receive information asynchronously, this is only true to the extent that no synchronizing clock is shared by the data sets on each end. But if the data sets are to talk to each other, their respective clocks must be very close to each other in frequency. With the TR1602B and similar UARTs, the clock rate is 16 times the rate at which data is sent (the baud rate). A very common rate is 300 baud, and so the clock frequency for 300 baud is 300 times 16 or 4,800 hertz (cycles per second). The baud rate is determined in advance, along with the number of data bits, parity bit, and number of stop bits.

While the UART waits for information, it checks the incoming line at a rate 16 times its baud rate. In other words, when the baud rate is 300 the UART checks its line 4,800 times a second. When the incoming line goes to a space condition, the UART waits 16 clock cycles, then samples the incoming line again. If the line is still low, the first bit is a logic zero. Since the number of bits in the character has been decided in advance, the UART repeats the above action for the number of bits in the character. If there is to be a parity bit, the UART checks for that, and then checks to make sure the line stays at mark for the preset number of stop bits.

If the parity is wrong, the tri-state status line that indicates a parity error is set. If the line did not stay at mark for the duration of the stop bit time, another tri-state status line is set. In the meantime, the data bits have been assembled in a register inside the UART and can be read out in the parallel format by the computer. The receive data bus has eight lines, so if the serial data format is less than eight bits, the unused parallel data bits are always set to logic low (zero). Serial data always goes out and comes in feet first—the first bit to be sent or received is always the least significant bit, and the last bit to be sent or received is always the most significant bit.

The UART itself is actually two units that can run entirely independent of each other. They can even run at different baud rates, though this is not often done. The transmit section can receive parallel data and send it out serially while the receive section takes in serial information and formats it for parallel output.

In fact, the clock generator in the TRS-80 serial interface has two outputs: one for the transmit clock and

one for the receive clock. The clock generator can be set to output two different frequencies. This IC is referred to as the baud rate generator (BRG).

I/O Ports

The serial interface board is controlled by the computer through the use of four input/output (I/O) ports. The TRS-80 has 256 I/O ports; the four used by the serial interface are 232, 233, 234 and 235 (ports E8, E9, EA and EB in hexadecimal).

Port 235 is used to transfer data bytes to and from the UART. Even though the data from the computer always has eight bits, the UART ignores all bits except those it is set to use.

The rest of the I/O ports pack a lot of status information into 8-bit bytes by using the individual bits as flags. A given flag will be set (a logic one) if something is going on, or not set (a logic zero) if not.

Port 234 is used to set up the operating conditions of the UART, and to send RTS and DTR flags to the data set. One of the operating conditions of the UART is parity enable. This bit is set if you want a parity bit sent with your data. The parity bit can be set to be a mark for an odd number of ones in the serial data or for an even number of ones. Parity odd/even is set by the computer to choose this condition. If the parity enable is not chosen, though, the parity odd/even flag has no effect.

Two bits of the byte output through port 234 select the number of data bits in the serial output. As I mentioned before, five, six, seven or eight bits can be selected. These two bits are known as word select bit 1 and word select bit 2. The number of stop bits is low for one stop bit and high for two stop bits. However, if you have selected a 5-bit word, the number of stop bits will be 1½, no matter how the number of stop bits is set.

The byte the computer receives from the input port 234 contains status information from the UART. The receive-data-available flag tells the computer that the UART has received a character of serial information and is ready to output it on the receive data bus. Transmit buffer empty tells the computer that the UART is ready to receive a parallel byte for serial transmission. The receiver-overrun flag indicates character was received before the UART could transfer a previous character to the receiver parallel buffer and set the receive-data-available flag. The receiver-framing-error flag is set if the UART detects a space condition on its serial input line when expecting the line to stay at mark for stop bits. If the parity option is in effect and the parity bit that the UART receives is not the parity it has calculated for the character preceding the parity bit, the receiver-parity-error flag will be set. See Table 2 for these flags and the data bit they are assigned to.

A set of switches on the serial interface board allows you to preset the operating parameters of the UART. The position of the switches can be sensed by reading information from port 233. The switches allow presetting parity odd/even, word length bit 1, word length bit 2, number of stop bits, and parity enable. In addition, three of the switches allow you to set the baud rate for 50–1200 for the transmit and receive clocks simultaneously. The baud-rate generator itself is capable of generating rates from 50–19,200 baud and accepts a separate 4-bit code for receive clock rate and transmit clock rate. The baud-rate generator is programmed by sending the appropriate byte out through output port 233. Data bits 4–7 contain the binary code for the transmit clock rate and data bits 0–3 contain the binary code for the receive clock rate.

The last port used by the serial interface board is I/O port 232. In the output mode, port 232 strobes the master reset pin of the UART.

Input 232 contains information concerning the status of the data set. CTS, DSR, CD and RI are sensed through this port. In addition, serial data bits from the data set will appear on data bit 0.

All the signals that go out to or come in from the RS-232C connector go through special ICs that either have TTL (5-volt logic) inputs and 20 mA current outputs or have 20 mA current inputs and TTL-level outputs (20 mA is the current specified by the RS-232C standard). ■

Howard Miller works at Siemens Corporation's Nuclear Medicine Division. He can be reached at 360 E. Winding River Drive, Dunwood, GA 30338.

Input Port

D7—Clear to send
D6—Data set ready
D5—Carrier detect
D4—Ring indicator
D3—Not used
D2—Not used
D1—Not used
D0—Serial data from RS-232C

Output Port

Master Reset for UART

Table 4. I/O Port 232 Configuration

Binary Code	Baud Rate
0000	50
0001	75
0010	110
0011	134.5
0100	150
0101	300
0110	600
0111	1200
1000	1800
1001	2000
1010	2400
1011	3600
1100	4800
1101	7200
1110	9600
1111	19,200

Table 5. Codes for Setting the Baud-rate Generator

TCS Heart of TEXAS COMPUTER SYSTEMS

LEADER in MAIL ORDER DISCOUNTS!
800 433-5184
Texas 817/274-5625

TANDON DRIVES

BARE DRIVES ONLY

40 track single sided	$199
80 track (dual sided 40)	$269
160 track (dual sided 80)	$389
5 ¼ inch THINLINE	$Call
8 inch THINLINE	$Call
Winchester Hard Drive	$Call

WE WILL NOT BE UNDERSOLD!

1 DRIVE / Single Cabinet

40 track single sided	$248
80 track (dual sided 40 track)	$329
180 track (dual sided 80 track)	$448

1 DRIVE / Double Cabinet

40 track single sided	$289
80 track (dual sided 40 track)	$368
180 track (dual sided 80 track)	$499

2 DRIVES / Double Cabinet

40 track single sided	$489
80 track (dual sided 40 track)	$639
180 track (dual sided 80 track)	$848

Drives in cabinets come assembled/tested with power supply. Order cable separately.

CORVUS HARD DISK

CORVUS HARD DISKS complete from $2695

Add 5, 10 or 20 megabytes of storage to your TRS80, Apple, Atari, Heath, Zenith, IBM, Intertec, S-100 and many others. One or several computers can share a hard disk. Get simultaneous access to data for multiple users. Available now at SUPER SAVINGS.

PRINTERS

TCS has the LOWEST PRICES on IN STOCK PRINTERS!

MATRIX PRINTERS

STAR MICRONICS	$399
NEC 8023A	$495
C. ITOH 8510 / TEC	$479
ANADEX 9501 w/2k buffer	$1295
LINE PRINTER (RS)	$Call
LINE PRINTER VI (RS)	$988
LINE PRINTER VII (RS)	$849
CENTRONICS 352 (200 cps)	$1795
CENTRONICS 353 (dual mode)	$2495
OKIDATA 82A	$429
OKIDATA 82A w/tractor	$479
OKIDATA 83A	$685
OKIDATA 84A (serial)	$1189
OKIDATA 84A (parallel)	$1089

HI SPEED BAND PRINTERS

CENTRONICS 6060 (standard)	$CALL
CENTRONICS (quietized)	$CALL

LETTER QUALITY PRINTERS

C.ITOH F-10 (40 cps)	$1595
DAISY WHEEL II (RS)	$1895
SMITH CORONA TP-1	$699
BROTHER / COMREX	$829
NEC 7730	$2375
NEC 3530	$1795

EPSON PRINTERS

EPSON MX80	$449
EPSON MX80 FT	$524
EPSON MX100 FT	$685

GRAPHTRAX PLUS comes free in Epsons!!

EPSON ACCESSORIES

GRAPHTRAX PLUS (bit image)	$78
MODEL I interface	$29
MODEL I interface/cable to keyboard	$85
MODEL III cable	$29
COLOR Computer interface/cable	$155
APPLE interface/cable	$89
ATARI cable (must have 850 int)	$36
IBM Personal Computer cable	$45
OSBORNE cable	$36
Serial RS232 2k buffered int.card	$109

EPSON PACK II

Screen oriented bit image graphics generator for the Model III. Create your own characters, symbols, etc. Documentation.

EPSON PACK II bought w/printer	$24.95
EPSON PACK II bought separately	$34.95
Update for EPSON PACK I	$7.50

B-TREE CUSTOM SOFTWARE FROM TCS

SCRATCHPAD

The BTREE Executive Reference Scratchpad allows the user to store all your phone numbers, business cards and what ever in a multi-key data base for immediate access by as many keys as you assign. Extremely easy to use. Allows 3 fields of Misc data

$39.95

LIBRARY

The BTREE Library program allows over 2000 books to be stored on a single double density data diskette in a efficient BTREE structured file for immediate access to your data by Author. Includes 4 build in reports. No sorting necessary for reports.

$39.95

MAIL LIST

The BTREE Mail List program allows over 2000 names to be stored on a single double density data diskette. Labels can be generated in zipcode order or by range of zipcodes and also by a code restriction if you wish.

$49.95

TRS-80

TCS MODEL III 48k 2 DISK

Systems come with 180 day TCS limited warranty.

$1695

With standard 40 track double density drives. Over 340,000 bytes. Includes TDOS.

$1995

With 2 dual headed 40 track dbl.density drives. Over 730,000 bytes. Includes TDOS.

Fully assembled and tested systems that are software competible and functionally identical to Radio Shack units sold at computer stores for $hundreds more.

- CONTROLLER BOARDS are high quality double sided epoxy boards with gold plated contacts.
- POWER SUPPLY is the finest switching type available.
- MOUNTING HARDWARE includes power and data cables.
- DISK DRIVES are Tandon, the same ones used by Radio Shack . . . 40 track, double density, with a 5 millisecond stepping rate.

TCS MODEL III DISK EXPANSION KITS

1	Controller, Power Supply, Mounting Hardware & Instructions	$279
2	Controller, Power Supply, Hardware & one 40 track Tandon drive	$478
3	Controller, Power Supply, Hardware & two 40 track Tandon drives	$677
3A	Kit 3 but with two 80 track Tandon drives (dual sided 40s)	$879
3B	Kit 3 but with two 180 track Tandon drives (dual sided 80s)	$1099

16k of high quality memory chips . . . $18 32k of high quality memory chips . . . $32

MODEL III SYSTEMS

Original 90 day manufacturer's warranty.

MODEL III 4k Level 1	$599
MODEL III 16k	$818
MODEL III 32k	$898
MODEL III 48k	$968
MODEL III 48k 2Dr/RS232	$1944

TCS MODEL III Systems use original RS hardware and quality TCS memory. 180 day limited warranty.

TCS MODEL III 16k	$799
TCS MODEL III 32k	$818
TCS MODEL III 48k	$848

COLOR COMPUTER

Original 90 day manufacturer's warranty.

16k Level 1	$306
16k Extended Basic	$398
32k Extended Basic	$CALL
Color Disk 0 $479	Disk 1 $349

TCS COLOR COMPUTERS use original RS hardware & TCS memory. 180 day warranty.

TCS 32k Extended Basic	$499
TCS 32k Memory Upgrade	$79
TCS Color Disk Drive 0	$448
TCS Color Disk Drive 1/2/3	$249

Model II . . Model 16 . . Accessories . . $CALL

TCS is an authorized TRS-80 dealer F701 in Brady, Texas

WIZARD'S TOUCH

Regularly $199 . Introductory Price 12/31/82

Program Development System $99

BELIEVE IT OR NOT, YOU CAN DEVELOP YOUR OWN PROGRAM IN 15 MINUTES!!!

The WIZARD'S TOUCH Program Development System is a superior program generator that can provide you with the program you3ve been looking for. It comes with it's own Disk Operating System, providing unlimited backups.

With WIZARD'S TOUCH, here's what you can do:

- Custom design your own special form for maximum ease of entering your data. You have total control over defining the number of characters assigned to each field and any other entry restrictions.
- Fully view and edit all fields at all times. No need to retype an entire field when editing. Just type over the mistakes, or insert or delete them.
- Create a BTREE structured file, allowing you to access your stored data rapidly and without sorting. Duplicate keys are allowed, as well as sub-keys.
- Enter data at full typing speed.
- Define your own special self-help or prompt information that will be displayed for any given field, by just touching one key.
- Expand your program to support up to 9 calculations for each data field. Results can be displayed in any field or in any defined display area. Results can also be saved to disk along with regular data fields. Data such as dates or prices can be formatted as desired.
- Generate your own reports with the built-in pre-defined or custom defined report generator. Just a few keystrokes can get you a report that can be alphabetized or ordered by any field. And it's all automatic.

The WIZARD'S TOUCH is so fast and easy to use. It makes other development systems of the market look like amateurs. No royalty necessary for resale of programs you develop on WIZARD'S TOUCH. It creates a stand-alone, fully commented, BASIC program, which you can easily modify.

Your purchase includes a free subscription to our WIZARD'S TOUCH newsletter, a quarterly publication with free programs, ideas for applications, updates and more.

Now available for MODEL I/III Coming soon for MODEL II, IBM and APPLE.

TEXAS COMPUTER SYSTEMS

P.O. Box 1327 Arlington, Texas 76004-1327 ~59

TECHNICAL ASSISTANCE 817/274-9221
ORDER STATUS 817/277-1913
TELEX/TWX/Easylink ELN 62100790

★ **800 433-5184**

No tax out of state. Texans add 5%. Prices subject to change at any time.

DEALER INQUIRIES INVITED

Ribbon Rewind

by Dan Keen and Dave Dischert

Learn how to rewind multistrike carbon ribbons for your Daisy Wheel II printer, and you can get seven extra printings from a cartridge.

Everybody wants to get the most they can out of the things for which they lay out their hard-earned cash.

Here is a trick for Daisy Wheel II owners that can save them hundreds of dollars a year in ribbon expenses.

The multistrike carbon ribbons (Radio Shack number 26-1419) for the Daisy Wheel printer are one-shot deals; they are designed to be used once and then thrown away. We have found that it is a very simple matter to rewind the ribbon and reuse it not once, not twice, but up to seven or eight times!

These ribbons cost $24.95 for a box of three at your local Radio Shack store. That figures out to about $8.32 each. Add sales tax, let's say five percent, and that brings the price up to $8.72 for one cartridge.

Now if you can get seven extra printings from a cartridge, that's a savings of $61.04 per cartridge or $183.12 per box!

The best news is that the quality of the second printing is almost identical to the first. Naturally, with each rewind of the ribbon the print will become slightly more faded. After about the fourth pass the print density reaches a point where you would not want to use it for formal letters. But it still remains fine for general computer use as well as line listings.

The Rewinding Procedure

First, remove the small belt for the top of the cartridge as seen in Photo 1. The cartridge itself pries apart very

Photos by George M. Keen

Photo 1

> ***The Key Box***
>
> **Any TRS-80**
> **Daisy Wheel II**

Photo 2

Photo 3

easily. Insert a small screwdriver into one of the large holes where the ribbon enters and exits the cartridge, as shown in Photo 2.

Once you have raised the corner, slip the screwdriver along the crack in the side (Photo 3). Continue to pry until you have gone completely around the circumference of the cartridge. The top will then pop off.

With the inside exposed, remove the clear thin circular sheet of plastic that sits on top of the supply reel. Carefully unlace the ribbon from around the gear and other guides. Position the ribbon so it goes directly from the take-up reel to the supply reel, bypassing any guides or wheels.

The supply reel has little hollow indentations, or bins, around its hub. This makes it easy to insert a screwdriver or ball point pen into any one of these bins and use it as a lever with which to rewind the ribbon. This is seen in Photo 4.

After a half dozen turns or so it is important to stop and push down on the ribbon as it goes onto the reel. It has a tendency to ride up as you wind. The ribbon cannot stick up or else it will be squeezed when the cover is put back on, preventing the ribbon from moving. Photo 5 demonstrates the best way to apply pressure, which is by pushing down on opposite sides of the reel using two or three fingers.

When the rewind is complete, the ribbon must be threaded again. In case you forget how you took the ribbon off, Fig. 1 shows the path of travel.

Upon completion of the rewind, replace the clear plastic disk on the supply reel. The top cover simply snaps back on. Place the belt back on top of the cartridge.

Using your fingers, turn the green manual knob that is used to take up slack. Look at the exposed ribbon as it passes through the open area and be sure it is moving along. If it isn't, pry the top off again and press hard on the ribbon several times. As we mentioned before, there is no extra room in that slim cartridge to accommodate any part of the ribbon that rises higher than the top of the reels.

You will become very proficient at this procedure after rewinding a few cartridges, and the whole operation will take only 10 or 15 minutes.

We have been using this technique for over a year and a half. We do heavy printing and, needless to say, have saved a small fortune in ribbons.

Occasionally you can only get a few passes from a cartridge. The ribbon itself is super thin and not very durable, since it is only designed for a single pass. Careless handling of the ribbon may cause it to break. It is usually impossible to recover a broken ribbon.

So start saving those used cartridges and cash in on the recycled market. ■

Dan Keen and Dave Dischert can be reached at RD1 Box 432, State Highway 83, Cape May Court House, NJ 08210.

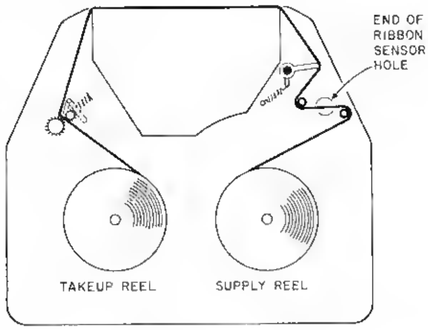


Figure 1

Photo 4

Photo 5

Homebrew Green Screen

by James Conroy

So you want to add a filter to your video screen, but you think they are too expensive? Follow these directions and build your own.

If you are tired of the harsh black-and-white video screen Radio Shack provides with the Models I, II and III, there is something you can do about it. Your options are to buy a $300 green phosphor monitor, buy a plastic video-screen filter for about $10, or build your own for less than $5!

Here's How

Most commercial video-filter products are made of acrylic resin-based plastics. This inexpensive plastic sheeting is available at your local hardware store or window-glazing supply house. Lucite, Plexiglass, Acrylite, and Polycast are pseudonyms for the same basic product and will do fine for a homebrew screen filter. You should choose a transparent green plastic sheet 1/8 inch thick (see Fig. 1).

Ask for a pre-cut piece of acrylic plastic 8½ by 11 inches. I purchased a sheet for $2.66.

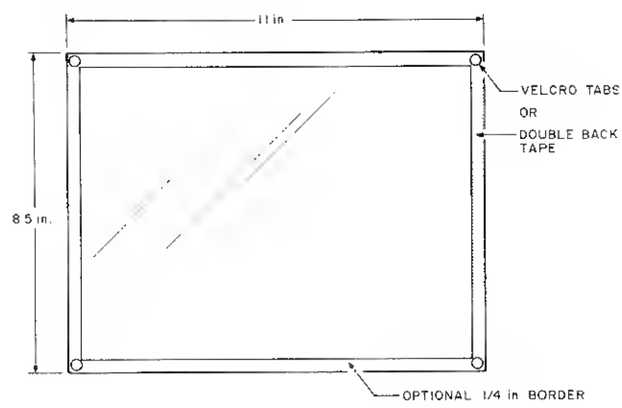


Figure 1

ACRYLITE	POLYCAST
GREEN 545-2	# 2092
GREEN 546-7	# 2082
GREEN 539-6	# 2414

A SAMPLE LISTING OF BRAND NAME PLASTICS AND PRODUCT NUMBER COLOR CODES

Figure 2

Mounting the Screen

Your plastic sheet is covered on both sides with protective adhesive paper; peel this off carefully. (This paper removes more easily in a warm environment than a chilly one.)

To copy the method used by video filter suppliers, find two narrow strips of pressure-sensitive adhesive tape (double-back carpet tape will do) and apply them to the vertical sides of your screen. (See Fig. 2.) It's a good idea to wipe the monitor and plastic sheet with alcohol to remove any grease before affixing the tape to the plastic. Press the screen to the monitor—that's all there is to it!

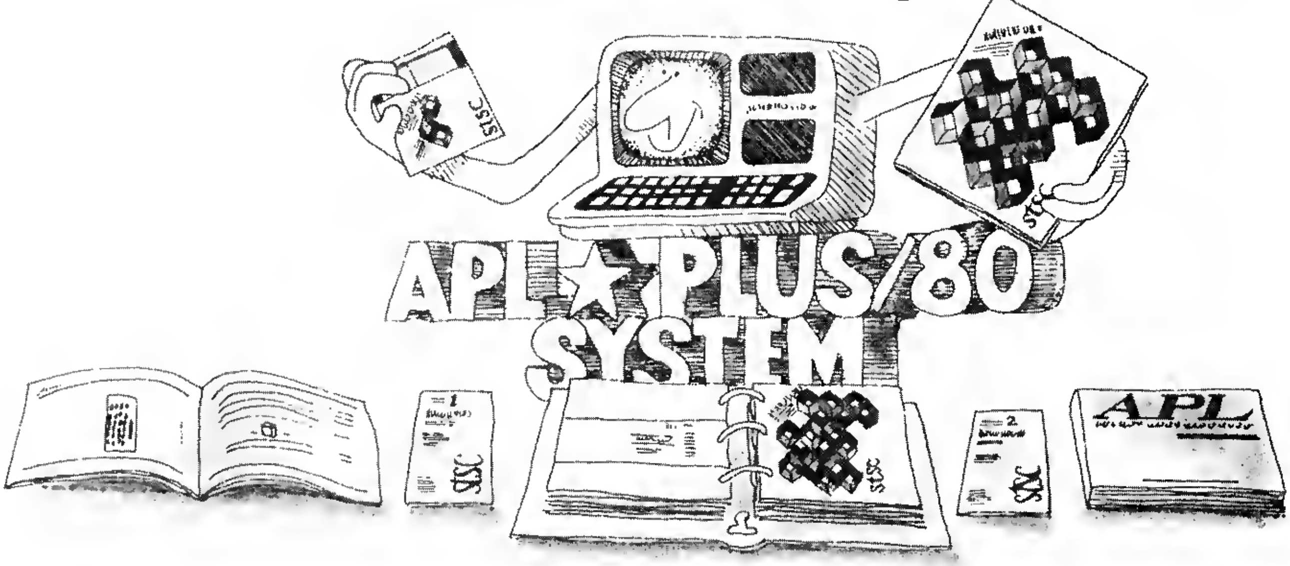
A Few Refinements

If you own a light pen, you will want to remove your filter from time to time. (The space between the plastic sheet and the video screen is too great for reliable light pen use.) A simple way to accommodate this is to use Velcro tabs instead of adhesive strips. Velcro holds firmly and yet allows you to place and remove your video screen at will. Velcro, used to secure carpet or electrical cords and garments, is available at hardware and sewing stores. Simply glue one tab at each corner of your plastic sheet and a corresponding tab on the monitor.

As a final refinement you may want to use silver or black paint for 1/4-inch margins on the plastic sheet. ■

James Conroy can be reached at 1227 Walnut St., Allentown, PA 18102.

A Practical and Educational Gift--
The APL★PLUS®/80 System.



All you get when you buy some software systems is a thin manual and a handshake for good luck. STSC's APL★PLUS/80 System is different. We give you a set of documentation that's so complete you won't need to read another thing to develop and run solutions on your TRS-80® Model III.

We know that our APL★PLUS/80 System can improve your productivity–if you know how to use its power. We start you off with a beginner's operation guide, if you're new to APL and to the TRS-80, and a beginner's tutorial, if you're familiar with the TRS-80 but not with the language. Then we include the most widely used APL textbook, so that once you get a taste for APL, you can study it in detail.

Challenge to BASIC

Build a data file with 15 records of 21 random numbers between 1 and 100 chosen without duplicates (in a record) and arranged in tables with 3 rows and 7 columns. Write a subroutine that reads a specified record and prints the table with row and column totals in fields 6 columns wide. Here is an APL solution:

```
     ∇ RANDOMTABLES;I
[1]    'RTABLES' ⎕FCREATE 10 ◇ I←1
[2]   L:(3 7⍴21?100)⎕FAPPEND 10 ◇ →(15≥I←I+1)⍴L
     ∇
     ∇ SHOWTABLE N;M
[1]    'I6' ⎕FMT M,[1]+/[1]M←M,+/M←⎕FREAD 10,N
     ∇
       RANDOMTABLES ◇ SHOWTABLE 12
     89     1    45    30    84    50   100   399
     52     8    93    13    99    40    77   382
     14    69    29    96     3    44    68   323
    155    78   167   139   186   134   245  1104
```

Here's your complete documentation package:

- Introduction to the APL★PLUS/80 System.
- Product Overview: APL★PLUS Systems.
- *APL★PLUS/80 Computer Operation User's Guide.*
- *APL is Easy!*
- *APL: An Interactive Approach,* Leonard Gilman and Allen J. Rose (Wiley, 1974).
- *APL★PLUS/80 Formatting User's Guide.*
- *APL★PLUS/80 Shared Files User's Guide.*
- *APL★PLUS/80 Programmer's Reference Manual.*
- Two APL★PLUS/80 Reference Cards.

We don't want to scare you away with our extensive documentation. You don't need to read all the manuals to write useful applications in APL, because APL is easy. APL is also fun and we're sure that when you know a little APL, you'll want to know a whole lot more–and *that's* why we're giving you the most complete set of documentation on the market.

If you'd like to use APL on your TRS-80 Model III to develop and deliver solutions faster and better for yourself or for your clients, send for the APL★PLUS/80 Application Development System. Mail in the coupon with your payment and we'll send you everything you need to run APL on your TRS-80 right away. If you'd like more details, check the box on the coupon and we'll send you our free information package which includes a detailed explanation of the Challenge to BASIC.

We're STSC, Inc., the leading supplier of APL software and services in the United States. Our APL★PLUS systems have been serving the business and professional world for more than 12 years.

APL★PLUS/80 runs under TRSDOS® 1.3 or LDOS® 5.1 on a 48K RAM TRS-80 Model III with two disk drives. The APL★PLUS/80 System comes with a custom APL-character ROM and a self-adhesive keyboard label set to convert your TRS-80 to the APL character set.

✓266

Attn: APL★PLUS/80 Distribution
STSC, Inc., 2115 East Jefferson Street
Rockville, Maryland 20852 (301) 984-5000 (orders only)

Yes, send me the APL★PLUS/80 System, postpaid.

☐ My check for $295 is enclosed. *Add sales tax where applicable.
☐ Charge my MasterCard Account # ______
Bank # ______
☐ Charge my VISA Account # ______
Expiration date: ______
*State sales tax applicable in MD (5%), FL (4%), and CT (7½%).

I'd like to know more about the APL★PLUS/80 System.

☐ Send me your free information package.

Name ______
Address ______
______
City ______ State ______ Zip ______
Phone (______) ______

✓See List of Advertisers on Page 563

Coping with Cassettes

by Richard Whitney

You will have a peaceful co-existence with your cassette recorder if you follow these tips and perform these maintenance procedures.

My problems with the cassette recorder began nearly five years ago. I figured I could save money if I used my existing audio cassette recorder instead of purchasing the cassette unit recommended for the TRS-80. Unfortunately, my cassette recorder drew more power than the control relay in the TRS-80 could handle. The relay contacts fused together causing the motor to run continuously. Radio Shack replaced the relay and I had learned my lesson.

Even after purchasing a new cassette recorder, using cassettes didn't become painless. However, I have finally learned to live with the cassette recorder and my cassette-based system is virtually trouble free.

Here are some tips that I have developed over the years that should help new and experienced TRS-80 users.

General Tape Tips

Use standard audio cassettes for file storage. Expensive certified computer cassettes are not worth the extra money. Most low-noise cassette tapes work just as well.

Use the short tapes: sizes C-10, C-20 or C-30. Longer tapes require extensive amounts of file searching and it's usually faster to change the tape than search for a file at the end of the tape.

Always *stop* the recorder when the tape motion stops for any significant length of time. When the tape is not moving, the capstan presses on one spot of the tape and will, after just a short time, cause a crease in the tape. This is probably the most common cause of data recovery errors. For example, always stop the recorder after loading a program from the tape and before executing it. It is easy to forget this after you have started the program.

Group program categories together on the same tape, i.e., games, utilities and so on. It organizes your tapes and makes retrieving files easier when you need them.

Use both sides of the tape instead of squeezing too many files on one side.

Reset the tape counter to zero when the tape is in the fully rewound position.

Label tapes with file grouping, group number (if more than one), baud rate (if your computer uses more than one rate), side of tape, and a description of each file. The description should include: tape counter starting number—first file copy, second copy; file name or description; file type—Basic or System; file identification used for access.

There is little room on the cassette to write all this information. The tape label should contain just the group name and number which matches the name on the tape box and the tape side information. Place a piece of paper in the tape box with the rest of the information. (See Table 1.)

Utilities Vol. 2

Baud Rate: 1500

Side 1.

Counter	File Description	Type	File ID
5, 35	Debugger	System	DBUG
70, 85	Renumber	Basic	"R"
105			

Side 2.

Table 1

Tape Recording Techniques

Never start recording a file at the very beginning of the tape even if the tape is leaderless. This portion of the tape is often stretched when the tape is rewound causing file recover errors. Allow the tape counter to advance to at least five.

Start files at tape counter multiples of five (or ten). This aids in keeping track of file positions and provides some spacing between files.

Make two copies of each file, one after the other placing the usual space between them. This helps ensure a good copy in case of a file read error. If the file is very important, you might

consider placing another copy on a separate tape.

Verify all copies of the file after recording them. Basic programs can be verified using the CLOAD? command. Most system program copiers usually provide the ability to verify files.

When recording over existing files, the new files are often of a different length than the file being overwritten. Pieces of the old file are thus left between the newly written files. To clean up any remains of the old files and make a cleaner, easier-to-read tape, first disconnect the motor control cable from the cassette recorder to control the tape motion manually. You can reconnect it after the file(s) is written.

Second, type in the command to write the file, but don't press enter yet. Third, press record and play together on the recorder, starting the recorder at a tape counter multiple of five or ten. If re-recording the first file on the tape, start the recorder on the clear plastic leader and advance it to five or ten to ensure that the beginning of the tape is clean.

Fourth, wait for the tape counter to advance beyond the selected count by approximately one count. Then start writing by pressing enter on the keyboard. The extra space after the indicated file starting count helps correct inaccuracies in the tape counter mechanism after fast forwarding or reversing the tape.

Fifth, when the file has been written on the tape and a prompt appears on the monitor, allow the recorder to continue to advance until the next multiple of five or ten. If the end of the file is near the next multiple, allow the count to proceed until the second count multiple. This provides an inter-file spacing that aids locating the beginning of files.

Sixth, stop the recorder and write down the counter value. That value is the start of the next file on the tape.

Finally, reconnect the motor control cable if this is the last file to be written, otherwise repeat the steps above for each file.

Tape Storage

Rewind the tape before storing it. Then only the tape leader is exposed and the tape stays clean.

When storing tapes, place them in the cassette tape boxes. It helps keep them cleaner and prevents the tape from becoming snagged.

Store tapes away from any magnetic devices, such as the power supply, tv monitor, and line printer. Such devices can affect data on the tapes.

Tape Recorder Maintenance

Clean the tape recorder heads regularly. Depending on use, this should vary from weekly to monthly. Use either a commercial tape recorder head cleaner, or 190-proof isopropyl (not rubbing) alcohol and a Q-tip.

Degausse tape heads after every three or four tape head cleanings. Degaussing removes magnetic buildup that interferes with recording and playback. You can purchase an inexpensive tape head degausser at most Radio Shack and many audio equipment stores.

I have presented some techniques for using the cassette tape recorder to store data for your home computer. These guidelines make the cassette recorder more reliable and even enjoyable to use. ■

Richard Whitney (105 O'Kelly Lane, Cary, NC 27511) is employed by Data General Corp.

✓See List of Advertisers on Page 563

Ultimate Joystick Interface

by Donald E. Michel and Art May

Wouldn't you rather use a joystick when you play games on your Mod I? This joystick interface is cheap, and easy to build and use.

The first item needed to build a joystick addition to your computer is the stick. Most sticks use potentiometers to read the paddle's position—voltage values vary as the stick is moved around. This arrangement works well, but is needlessly complex. The computer only wants ones and zeros anyway.

Atari manufactures a joystick that simply depresses a switch anytime the stick is fully thrown into a particular position. There had to be a way to make this stick fit my requirements (inexpensive, no complicated hardware modifications), and indeed there is.

The Circuitry

The schematic in Fig. 1 shows one

> *The Key Box*
>
> **Basic Level II**
> **Model I**

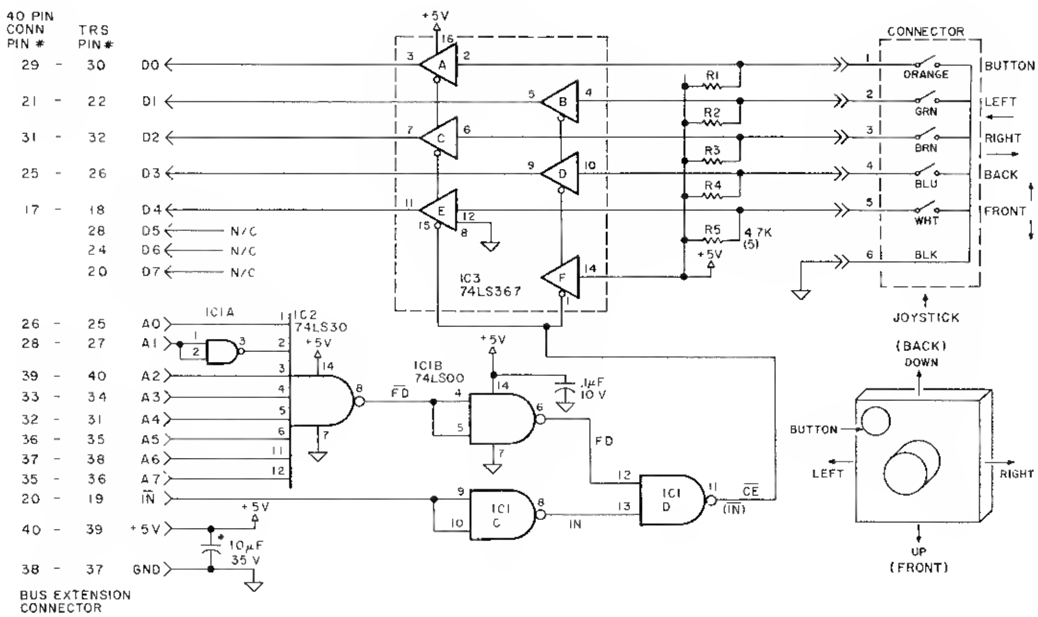


Fig. 1. Joystick Interface

improvement over some other designs: our port (0FDH) is properly decoded by the 74LS30 IC. This technique makes the interface essentially transparent to the rest of system, avoiding hang-ups if an inadvertent call to a port is made.

The address line A1 is inverted by pins 1 and 2 of the 74LS00 (IC 1), while the balance of the address lines feed directly to the 74LS30. The resulting signal is inverted once again and, when coupled with the In line going to pin 13, creates all the logic necessary to selectively fire the buffers of IC3. This in turn gates the existing bit pattern of the joystick to the TRS-80's bus. Simple?

It is, if you've ever done this sort of thing before. Still, one or two inconsistencies rear their ugly heads. No big deal, but you have to be careful.

One notable oddity is the fact that no 40-pin card edge connector in the world has pin designations corresponding to the TRS-80's pin numbers. The TRS-80 numbering is always one digit different than the connector mating with it, thus the discrepancy (see Fig. 2 for proper orientation of TRS-80 and connector pin designations).

Another minor hassle concerns the power needed to run the thing. If you do not have an expansion interface, power your unit from the +5 and ground pins available from the bus extension at the keyboard. The expansion interface does not provide +5 volts, but improvising is easy. Cut the trace between pins 37 and 39 (pin 39 is an unnecessary ground, see Fig. 3). Then jumper +5 volts from the reverse side of the PC board (Fig. 4). The LS chips consume very little power, so don't worry about burdening your power supply.

Construction

The easiest approach is to wire-wrap the interconnections after mounting the parts on a suitable PC board. I chose the parts listing in Fig. 1 for ease of assembly and to keep the unit's size to a minimum. There are not many connections to be made and construction goes quickly.

Software

How easy it is to develop routines to interface your joystick with your programs depends on your skill as a programmer and what it is you want to do.

As a simple example, and to test your newly completed interface, type in the Basic program in Listing 1. An asterisk should appear and move in directions determined by the paddle. Pressing the fire button causes the word Fire to blink on the screen for as long as it is depressed.

An INP command examines the bit pattern on the data lines. The A variable takes on the complement of the input port ANDed with 31 to mask the lower five bits. A is checked for set bit positions. Depending on the outcome of the check, it either processes Fire or moves the asterisk. Lines 120–140 keep you from cruising off the screen, while line 150 keeps the screen from scrolling. Line 190 does the actual printing, then sends execution back to the polling routine at line 60.

The program is given as an example; the interface is certainly not limited to such trivia. The Atari paddle can operate with two of its internal switches depressed at the same time by moving the stick diagonally. Diagonal movement is provided by sequentially checking each bit position.

Once you get the hang of using Basic's INP command and can successfully read your joystick's port, you are free to do whatever you want. And you are not stuck with Basic, either.

The joystick subroutine can be POKEd into memory. For the ultimate in realism, patch machine-language routines into existing arcade-style games. The results are amazing. See Program Listing 2 for a general Z80 routine.

Listing 2 is a typical machine-language module for computer control via the joystick. Line 10 copies the input port to the A register. Line 20 looks for

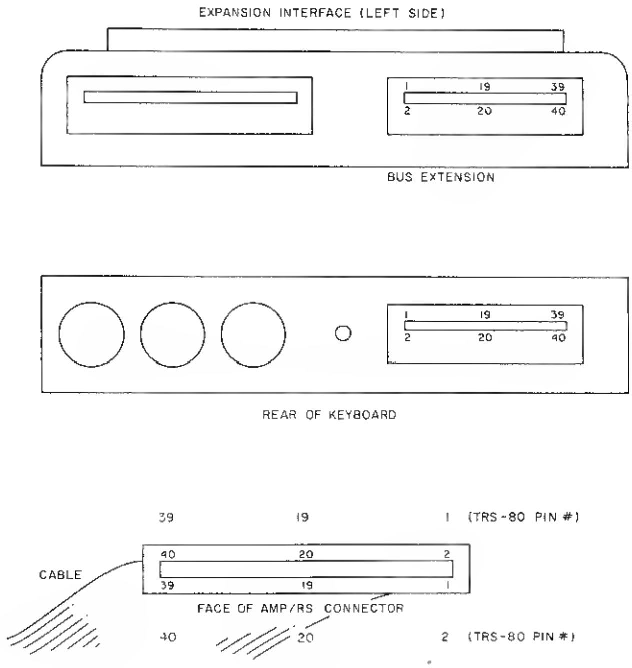


Figure 2

```
10 CLS
20 DEFINT A-Z
30 X1=90
40 Y=7
50 X=30
60 A=NOT(INP(253)AND 31)
70 IF A AND 1 THEN PRINT@0,"FIRE":FOR I=1TO100:NEXT:PRINT@0,"
80 IF A AND 2 THEN X=X-1
90 IF A AND 4 THEN X=X+1
100 IF A AND 8 THEN Y=Y+1
110 IF A AND 16 THEN Y=Y-1
120 IF X<0 THEN X=0
130 IF X>63 THEN X=63
140 IF Y<0 THEN Y=0
150 IF Y>15 THEN Y=15
160 IF Y=15 AND X=63 THEN X=62
170 X2=64*Y+X
180 PRINT@X1," ";
190 PRINT@X2,"*";
200 X1=X2
210 GOTO60
```

Program Listing 1

joystick activity (0FFH indicates no joystick activity). Line 30 continues the program if all bits are set. Line 40 rotates the lowest bit into the condition code register. Line 50 calls the button response routine in the main program. The remainder of the module successfully checks for zero bits and calls the proper response routine in the main program. Line 140 continues the user's program.

Commercially available joysticks will do everything this one does. But you have to shell out considerably more money for a store-bought unit than parts for this one. This interface is easily adapted to any program and is easy to tack onto your own graphics creations.

Hopefully, you will never again be obliterated by unscrupulous Klingons while frantically fumbling for the up arrow. Good luck! ■

Donald E. Michel is Chief Engineer at T. J. Electronics. Art May is Senior Electronics Specialist at C. R. C. Wireline.

```
10   START   IN     A,(0FDH)
20           CP     0FFH
30           JP     Z,CONT
40           RRA
50           CALL   NC,BUTTON
60           RRA
70           CALL   NC,LEFT
80           RRA
90           CALL   NC,RIGHT
100          RRA
110          CALL   NC,UP
120          RRA
130          CALL   NC,DOWN
140          JP     CONT
```

Program Listing 2

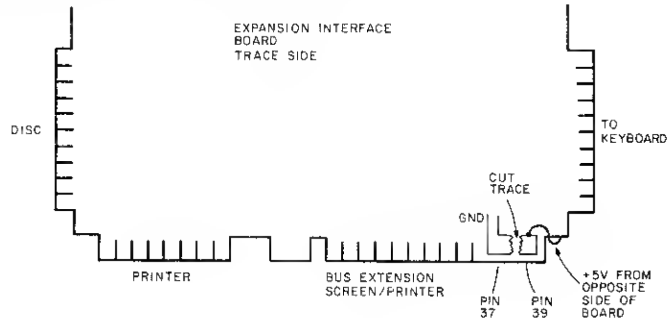


Figure 3

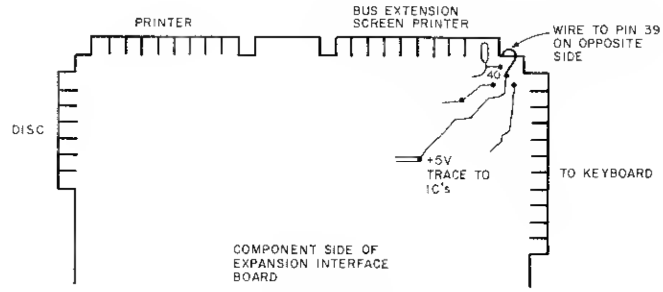


Figure 4

HOW TO USE YOUR EPSON WITHOUT WASTING COMPUTER TIME:

Your computer is capable of sending data at thousands of characters per second but the Epson can only print 80 characters per second.

This means your computer is forced to wait for the printer to finish one line before it can send the next. A waste of valuable time.

THE NEW MICROBUFFER™ ACCEPTS DATA AS FAST AS YOUR COMPUTER CAN SEND IT.

Microbuffer stores the data in its own memory buffer and then takes control of the printer. This frees your computer for more productive functions.

PARALLEL OR SERIAL.

Microbuffer model MBP-16K is a Centronics-compatible parallel interface with 16,384 bytes of on-board RAM for data buffering.

The MBS-8K is a full-featured RS-232C serial interface with both hardware and software (X-On/X-Off) handshaking, baud rates from 300 to 19,000 and an 8,192 byte RAM buffer.

SIMPLY PLUG IT IN.

Either model fits the existing auxiliary interface connector inside the Epson MX-80, MX-80 F/T or MX-100 without modification, and is compatible with standard Epson cables and printer control software, including GRAFTRAX-80.

JUST $159.00*

When you think how much time Microbuffer will save, can you afford *not* to have one? Call us for your nearest dealer.

PRACTICAL PERIPHERALS, INC.
31245 LA BAYA DRIVE, WESTLAKE VILLAGE, CA 91362 • (213) 991-8200

11

galaxy of features

A GALAXY of features makes the **LNW80** a remarkable computer. As you explore the **LNW80**, you will find the most complete, powerful, ready to run, feature-packed personal and business computer ever made into one compact solid unit.

MODEL I COMPATIBILITY – The **LNW80** is fully hardware and software compatible with the Model I. Select from a universe of hardware accessories and software – from VisiCalc® to space games, your **LNW80** will launch you into a new world of computing.

FULLY LOADED – A full payload includes an on-board single and double density disk controller for 5 ¼" and 8" single or double sided disk drives, RS232C communications port, cassette and parallel printer interfaces are standard features and ready to go. All memory is fully installed – 48K RAM, 16K graphics RAM and 12K ROM complete with Microsoft **BASIC**.

QUALITY CONSTRUCTION – Instrumentation quality construction sets **LNW80** computers apart from all the rest. Integrated into the sleek solid steel case of the **LNW80** is a professional 74-key expanded keyboard that includes a twelve key numeric keypad.

HIGH RESOLUTION GRAPHICS & COLOR – The stunning 480 X 192 resolution gives you total display control – in color or black and white. The choice of display formats is yours: 80, 64, 40 and 32 columns by 24 or 16 lines in any combination of eight colors.

PERFORMANCE – Lift-off with a 4MHz Z80A CPU for twice the performance. The **LNW80** outperforms all computers in its class.

Our down to earth price won't send you into orbit.

LNW Research Corp.

[illegible] WALNUT [illegible], CA [illegible]

[illegible] (714) 544-[illegible]

~30

[illegible]